# Datasheet for Telink 2.4GHz RF SoC TLSR8367

**DS-TLSR8367-E2**

**Ver 0.8.1**

**2020/11/17**

Under NDA

**Keyword:**

2.4GHz; Features; Package; Pin layout; Memory; MCU; Working modes; Wakeup sources; RF Transceiver; Baseband; Clock; Timers; Interrupt; Interface; PWM; QDEC; ADC; PGA; AES; Electrical specification

**Brief:**

This datasheet is dedicated for Telink 2.4GHz RF SoC TLSR8367. In this datasheet, key features, working mode, main modules, electrical specification and application of the TLSR8367 are introduced.

TELINK SEMICONDUCTOR

**Published by**
**Telink Semiconductor**

**Bldg 3, 1500 Zuchongzhi Rd,**
**Zhangjiang Hi-Tech Park, Shanghai, China**

**© Telink Semiconductor**
**All Right Reserved**

**Legal Disclaimer**

Telink Semiconductor reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein or in any other disclosure relating to any product.

Telink Semiconductor does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights, nor the rights of others

The products shown herein are not designed for use in medical, life-saving, or life-sustaining applications. Customers using or selling Telink Semiconductor products not expressly indicated for use in such applications do so entirely at their own risk and agree to fully indemnify Telink Semiconductor for any damages arising or resulting from such use or sale.

**Information:**

For further information on the technology, product and business term, please contact Telink Semiconductor Company (www.telink-semi.com).

For sales or technical support, please send email to the address of:

telinkcnsales@telink-semi.com

telinkcnsupport@telink-semi.com

Shared under NDA

## Revision History

| Version | Major Changes | Date | Author |
|---|---|---|---|
| 0.8.0 | Preliminary release | 2019/8 | LWF, SY, YHL, JF |
| 0.8.1 | 1.Update deep and suspend current in 1.2.3 and Table 14-3<br>2. Update sensitivity data in 1.2.2 and Table 14-9 | 2020/11 | LWF, YLJ,YH |

Shared under NDA

# 1 Table of contents



Shared under NDA

Shared under NDA

Shared under NDA

Shared under NDA

Shared under NDA

# 1 Overview

The RoHS-compliant TLSR8367 is dedicated to 2.4GHz RF System-On-Chip solution, such as wireless keyboard, wireless mouse, non-audio remote control applications, and etc.

## 1.1 Block diagram

The TLSR8367 integrates a power-balanced 32-bit proprietary MCU, a high-performance 2.4GHz Radio, 8kB SRAM, 16kB OTP, a general-purpose ADC, a quadrature decoder (QDEC), 5-channel PWM, flexible I/O interfaces and other peripheral blocks required for 2.4GHz RF System-On-Chip solution. Few external components are needed to satisfy customers' ultra-low cost requirements.

The system's block diagram is as shown in Figure 1-1:

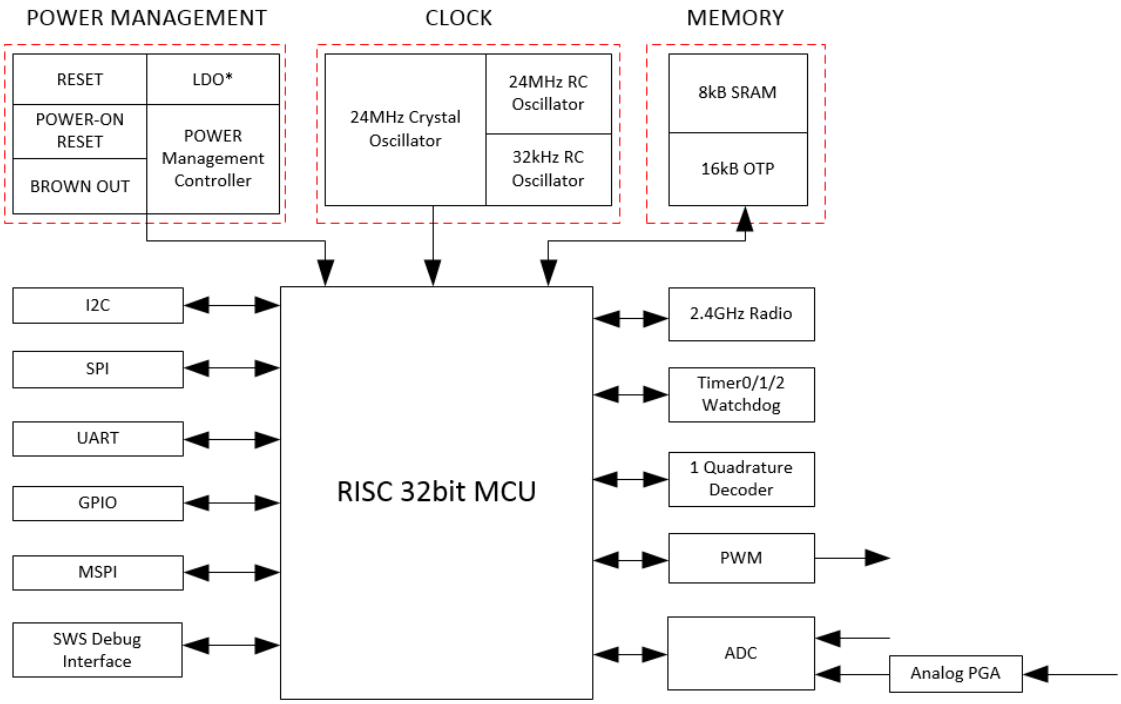



Figure 1- 1 Block diagram of the system

***Note:** The internal LDO regulators serve to supply power for 1.8V digital core and analog modules in Active/Idle/Suspend mode.

## 1.2 Key features

### 1.2.1 General features

General features are as follows:

1) 32-bit proprietary microcontroller

   ✧ Better power-balanced performance than ARM M0

NDA

- ✧ Instruction cache controller with 2kB cache RAM memory
- ✧ Maximum running speed up to 48MHz

2) Memory architecture
   - ✧ 16kB OTP
   - ✧ 8kB SRAM
3) Supports 2.4GHz proprietary protocols
4) RTC and other timers
   - ✧ Clock source of 24MHz/32kHz RC oscillator, 24MHz crystal oscillator
   - ✧ Three general 32-bit timers, with four selectable modes in active mode
   - ✧ Watchdog timer
   - ✧ A low-frequency 32kHz timer available in low power mode
5) Digital and analog interfaces
   - ✧ Up to 31/15/9 GPIOs depending on package option
   - ✧ Configurable pull-up or pull-down resistors
   - ✧ I2C Master/Slave
   - ✧ SPI Master/Slave
   - ✧ Memory SPI
   - ✧ UART interface with hardware flow control
   - ✧ SWS (Single Wire Slave) interface for debugging
   - ✧ One quadrature decoder (QDEC)
   - ✧ Up to 5-channel PWM output
   - ✧ IR transmitter with DMA support
   - ✧ Up to 10-channel (only GPIO input) ADC with 10.5 ENOB
   - ✧ 4-channel differential PGA.
6) Hardware AES and random number generator
7) Firmware encryption: support software signature based on OTP
8) Operating temperature range: -40℃~+85℃
9) Package:
   - ✧ 16-pin SOP16L_9.9x6mm, TLSR8367EP16

### 1.2.2 RF Features

RF features include:

1) 2.4GHz RF transceiver in worldwide 2.4GHz ISM band
2) 2.4GHz proprietary 2Mbps/1Mbps/500kbps/250kbps mode with Adaptive Frequency Hopping support
3) Rx Sensitivity: -90dBm @ 2.4GHz 1Mbps mode, -86.5dBm @ 2.4GHz 2Mbps mode

Shared under NDA

4) Tx Output power: +7dBm
5) 50 Ω matched single-pin antenna input
6) RSSI monitoring with +/-4dB accuracy

### 1.2.3 Features of power management module

Features of power management module include:

1) Power supply of 1.9V~4.3V (QFN package) / 1.9V~3.6V (SOP16 package)
2) Battery monitor for low battery voltage detection
3) Brownout detection/shutoff and Power-On-Reset
4) Multiple-power-state to optimize power consumption
5) Low power consumption
   - Transmitter mode current: 14.5mA @ 0dBm power, 25mA @ 8dBm power
   - Receiver mode current: 13.6mA
   - Suspend mode current:
     - IO wakeup: 6.8uA
     - 32kHz RC wakeup: 8uA
   - Deep sleep mode current:
     - internal 32kHz RC OSC off: 1.9uA
     - internal 32kHz RC OSC on: 3uA

Shared under NDA

## 1.3 Typical application

The TLSR8367 can be applied to 2.4GHz RF System-On-Chip solutions.

Its typical applications include, but are not limited to:

- Wireless keyboard
- Wireless mouse
- Non-audio remote control

## 1.4 Ordering information

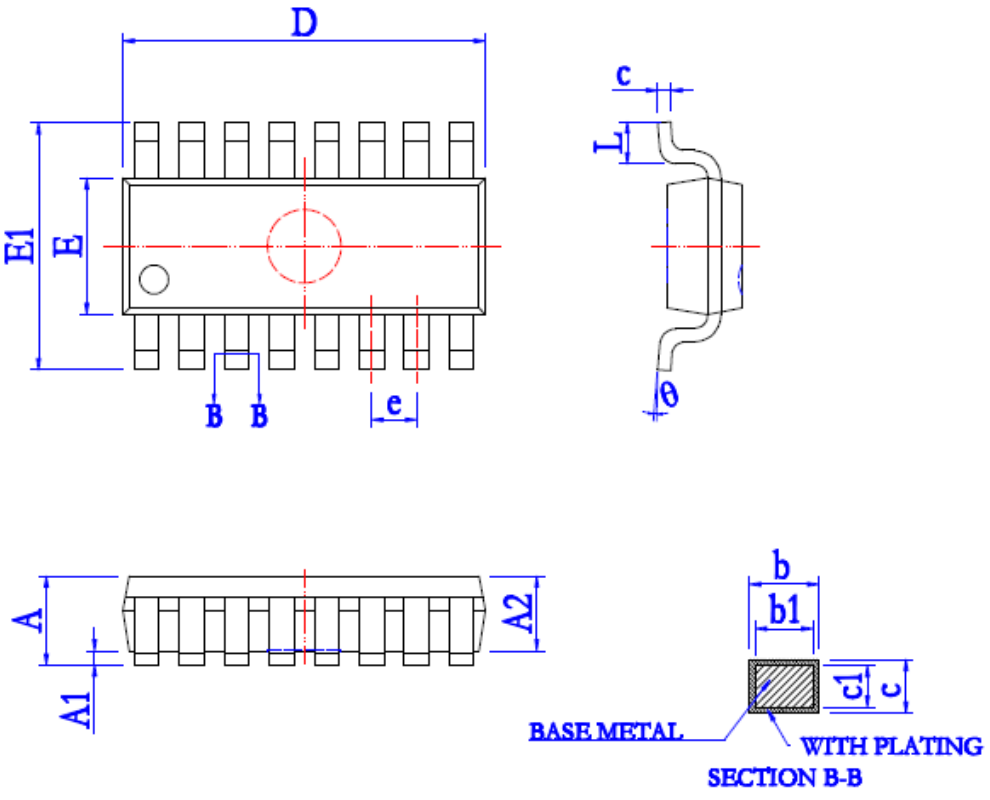
Table 1- 1TLSR8367 ordering information

| Product Series | Package Type | Temperature Range | Product Part No. | Packing Method | Minimum Order Quantity |
|---|---|---|---|---|---|
| TLSR8367 | 16-pin SOP16L_ 9.9x6mm | -40℃ ~ +85℃ | TLSR8367EP16 | Tube | 5000 |
| TLSR8367E02 | 16-pin SOP16L_ 9.9x6mm | -40℃ ~ +85℃ | TLSR8367E02EP16 | Tube | 5000 |

*Note: Packing method “TR” means tape and reel.

## 1.5 Package

Package dimension for the TLSR8367EP16/TLSR8367E02EP16 is shown as below.




Confidential under NDA

| SYMBOL | MILIMETER | |
|---|---|---|
| | MIN | MAX |
| A | 1.350 | 1.750 |
| A1 | 0.100 | 0.250 |
| A2 | 1.350 | 1.550 |
| b | 0.330 | 0.510 |
| b1 | 0.320 | 0.500 |
| c | 0.170 | 0.250 |
| c1 | 0.160 | 0.240 |
| D | 9.800 | 10.200 |
| E | 3.800 | 4.000 |
| E1 | 5.800 | 6.200 |
| e | 1.270BSC | |
| L | 0.400 | 0.800 |
| θ | 0° | 8° |
| L/F Carrier Dimension (mil) | 134*91 | |

Figure 1- 2 Package dimension for TLSR8367EP16 (Unit: mm)

Shared under NDA

## 1.6 Pin layout

### 1.6.1 Pin layout for TLSR8367EP16

The figure below shows pin assignment for the TLSR8367EP16.

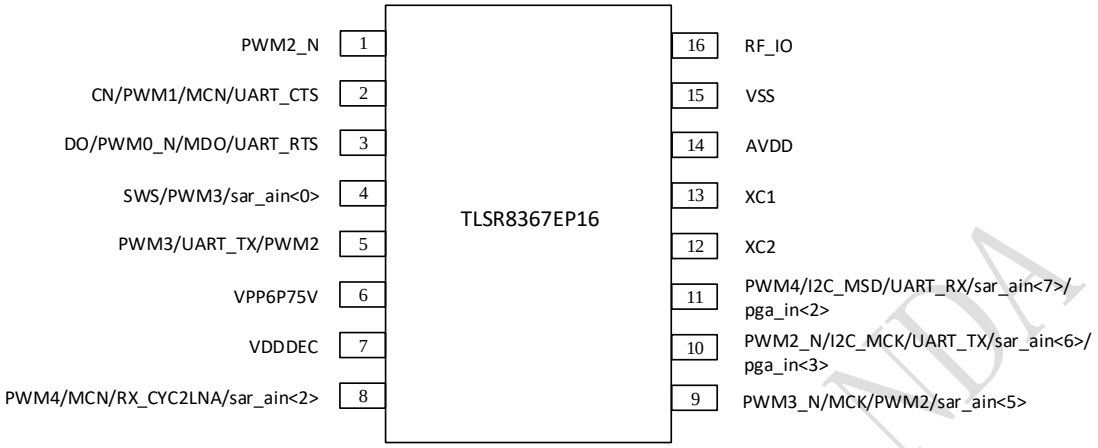



Figure 1- 3 Pin assignment for TLSR8367EP16

Functions of 16 pins for the TLSR8367EP16 is described in the table below:

Table 1- 2 Pin functions for TLSR8367EP16

| SOP16L_10X6 | | | |
|---|---|---|---|
| **No.** | **Pin Name** | **Pin Type** | **Description** |
| 1 | PWM2_N / PC[1] | Digital I/O | PWM2_N output / GPIO PC[1] |
| 2 | CN / PWM1 / MCN / UART_CTS / PC[2] | Digital I/O | SPI Chip Select / PWM1 output / SPI Master chip select (active low) / UART_CTS / GPIO PC[2] |
| 3 | DO / PWM0_N / MDO / UART_RTS / PC[3] | Digital I/O | SPI data output / PWM0_N output / SPI Master data output / UART_RTS / GPIO PC[3] |
| 4 | SWS / PWM3 / sar_ain<0> / PC[7] | Digital I/O | Single wire slave / PWM3 output / SAR ADC input / GPIO PC[7] |
| 5 | PWM3 / UART_TX / PWM2 / PD[2] | Digital I/O | PWM3 output / UART_TX / PWM2 output / GPIO PD[2] |
| 6 | VPP6P75V | PWR | for OTP program 6.75V power supply |
| 7 | VDDDEC | PWR | Digital LDO output |
| 8 | PWM4 / MCN / RX_CYC2LNA / sar_ain<2> / PB[0] | Digital I/O | PWM4 output / SPI Master chip select (active low) / Control external LNA / SAR ADC input / GPIO PB[0] |
| 9 | PWM3_N / MCK / PWM2 / sar_ain<5> / PB[3] | Digital I/O | PWM3 inverting output / SPI Master clock / PWM2 output / SAR ADC input |

Shared under NDA

| SOP16L_10X6 | | | |
|---|---|---|---|
| **No.** | **Pin Name** | **Pin Type** | **Description** |
| | | | / GPIO PB[3] |
| 10 | PWM2_N / I2C_MCK / UART_TX / sar_ain<6> / pga_in<3> / PB[4] | Digital I/O | PWM2 inverting output / I2C Master clock / UART_TX / SAR ADC input / PGA input / GPIO PB[4] |
| 11 | PWM4 / I2C_MSD / UART_RX / sar_ain<7> / pga_in<2> / PB[5] | Digital I/O | PWM4 output / I2C Master serial data / UART_RX / SAR ADC input / PGA input / GPIO PB[5] |
| 12 | XC2 | Analog I/O | 24MHz crystal input+ |
| 13 | XC1 | Analog I/O | 24MHz crystal input- |
| 14 | AVDD | PWR | 3.3V supply |
| 15 | VSS | GND | ground for the whole chip |
| 16 | RF_IO | Analog I/O | RF signal input/output (antenna) |

Shared under NDA

### 1.6.2 Notes

1) All digital IOs including PA[0] ~ PD[3] can be used as GPIOs and have configurable pull-up/pull-down resistor.

2) I2C and SPI Master/Slave pins can be configured independently.

   - ✧ Pins marked with I2C_MCK and I2C_MSD can be configured as I2C Master clock and serial data.
   - ✧ Pins marked with I2C_CK and I2C_SD can be configured as I2C Slave clock and serial data.
   - ✧ Pins marked with MCN, MDO, MDI and MCK can be configured as SPI Master chip select (active low), data output, data input and clock.
   - ✧ Pins marked with CN, DO, DI and CK can be configured as SPI Slave chip select (active low), data output, data input and clock.

3) RX_CYC2LNA & TX_CYC2PA: control enabling external PA/LNA. Please refer to **section 3.1** Block diagram.

4) UART with hardware flow control: UART_TX, UART_RX, UART_CTS, UART_RTS.

5) ADC input: PC[7], PA[7], PB[0], PB[1], PB[3], PB[4], PB[5]. Please refer to **section 10 SAR ADC**.

6) Analog PGA input: PB[4], PB[5]. Please refer to **section 11 PGA**.

7) Pin drive strength: PA[0] ~ PD[3] support drive strength up to 8mA (8mA when “DS”=1, 4mA when “DS”=0). Please refer to section **7.1.1 Basic configuration** for the corresponding “DS” register address and the default setting.

Shared under NDA

# 2 Memory and MCU

## 2.1 Memory

The TLSR8367 embeds 8kB SRAM and 16kB OTP.

### 2.1.1 SRAM/Register

SRAM/Register memory map is shown as follows:


Figure 2- 1 Physical memory map

Register address: 0x800000 ~ 0x807FFF;

8kB SRAM address: 0x808000 ~ 0x809FFF.

Both register and 8kB SRAM address can be accessed (read or write) via debugging interface (SPI/I2C, SWS interface).

### 2.1.2 OTP

OTP address mapping is configurable.

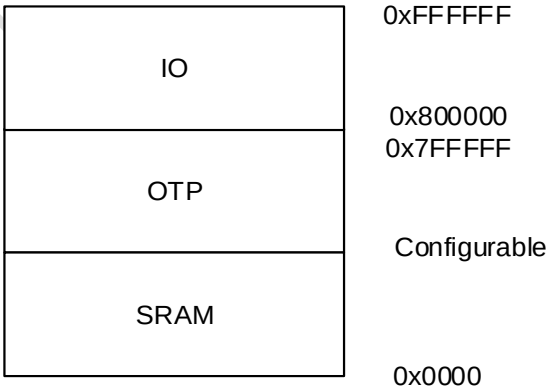



Figure 2- 2 MCU memory map

Shared under NDA

## 2.2 Firmware encryption

The TLSR8367 supports Bootloader-based firmware encryption/decryption.

The firmware can be encrypted using a customer-provided security key. The customer security key is written into a specified location within the internal OTP (i.e. the last 4-bytes of the OTP), and becomes unreadable. Any attempt to read the key will only result in either all 1's or all 0's.

The encrypted firmware can be generated based on the plaintext firmware and the customer security key. The customer can burn the security key into the obscured memory area and also the encrypted firmware into the OTP.

The firmware is readable by all, but appears as garbled binaries to 3rd party.

## 2.3 MCU

The TLSR8367 integrates a powerful 32-bit MCU developed by Telink. The digital core is based on 32-bit RISC, and the length of instructions is 16 bits; four hardware breakpoints are supported.

## 2.4 Working modes

The TLSR8367 has four working modes: Active, Idle, Suspend and Deep Sleep. This section mainly gives the description of every working mode and mode transition.

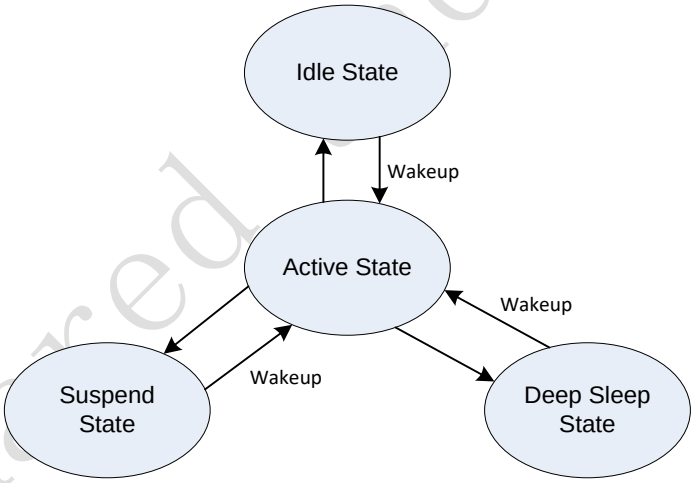



Figure 2- 3 Transition chart of working modes

### 2.4.1 Active mode

In active mode, the MCU block is at working state, and the TLSR8367 can transmit or receive data via its embedded RF transceiver. The RF transceiver can also be powered down if no data transfer is needed.

### 2.4.2 Idle mode

In Idle mode, the MCU block stalls, and the RF transceiver can be at working state or be

Shared under NDA

powered down. The time needed for the transition from Idle mode to Active mode is negligible.

### 2.4.3 Power-saving mode

For the TLSR8367, there are two kinds of power-saving modes: suspend mode and deep sleep mode. The two modes have similar transition sequences but different register settings. For 1.8V digital core, it's still provided with the working power by 1.8V LDO in suspend mode; while in deep sleep mode, the 1.8V LDO will be turned off, and the digital core is powered down.

In suspend mode, the RF transceiver is powered down, and the clock of the MCU block is stopped.

While in deep sleep mode, both the RF transceiver and the MCU block are powered down with only power management block being active.

Deep sleep mode includes two sub-modes, including deep sleep with SRAM retention and deep sleep without SRAM retention.

Table 2- 1Deep sleep sub modes

| Deep Sleep Sub-Mode / Characteristic | Deep sleep with SRAM retention | Deep sleep without SRAM retention |
|---|---|---|
| Wakeup time to Active mode | Shorter than deep sleep without retention, almost same as Suspend | 1ms |
| 8kB retention SRAM (with retention in deep sleep) | full | off |

***Note:**

1) "full": Full speed. In Active, Idle and Suspend mode, the 8kB retention SRAM is powered on and work normally (can be accessed); in Deep sleep with SRAM retention, the retention SRAM is powered on, however, the contents of the retention SRAMs can be retained and cannot be accessed.
2) "off": The retention SRAM is powered down in Deep sleep without SRAM retention.

$T_{SP2A}$: the transition time needed for the TLSR8367 to enter the active mode from suspend mode.

$T_{DS2A}$: the transition time needed from deep sleep mode to active mode.

For the tested values of the "$T_{SP2A}$" and "$T_{DS2A}$", please refer to **section 14.4 General characteristics**.

Shared under NDA

Table 2- 2 Retention analog registers in deep sleep

| Address | R/W | Description | Default value |
|---|---|---|---|
| afe_0x34 | R/W | buffer, watchdog reset/software reset clean | 0x00 |
| afe_0x35 | R/W | buffer, watchdog reset/software reset clean | 0x00 |
| afe_0x36 | R/W | buffer, watchdog reset/software reset clean | 0x00 |
| afe_0x37 | R/W | buffer, watchdog reset/software reset clean | 0x00 |
| afe_0x38 | R/W | buffer, watchdog reset/software reset clean | 0x00 |
| afe_0x39 | R/W | buffer, watchdog reset/software reset clean | 0xff |
| afe_0x3a | R/W | buffer, only power on reset clean | 0x00 |
| afe_0x3b | R/W | buffer, only power on reset clean | 0x00 |
| afe_0x3c | R/W | buffer, only power on reset clean | 0x00 |
| afe_0x3d | R/W | buffer, only power on reset clean | 0x00 |
| afe_0x3e | R/W | buffer, only power on reset clean | 0x00 |
| afe_0x3f | R/W | buffer, only power on reset clean | 0x7f |

Analog registers (afe_0x34 ~ afe_0x3f) as shown above are retained in deep sleep mode and can be used to store program state information across deep sleep cycles.

- ✧ Analog registers afe_0x3a~ afe_0x3f are non-volatile even when chip enters deep sleep or chip is reset by watchdog or software, i.e. the contents of these registers won't be changed by deep sleep or watchdog reset or chip software reset.
- ✧ Analog registers afe_0x34~ afe_0x39 are non-volatile in deep sleep, but will be cleared by watchdog reset or chip software reset.
- ✧ After POR (Power-On-Reset), all registers will be cleared to their default values, including these analog registers.

User can set flag in these analog registers correspondingly, so as to check the booting source by reading the flag.

For chip software reset, please refer to **section 2.5 Reset**.

Shared under NDA

## 2.5 Reset

The chip supports three types of reset methods, including POR (Power-On-Reset), watchdog reset and software reset.

1) POR: After power on, the whole chip will be reset, and all registers will be cleared to their default values.

2) Watchdog reset: A programmable watchdog is supported to monitor the system. If watchdog reset is triggered, registers except for retention analog registers afe_0x3a~ afe_0x3f will be cleared.

3) Software reset: It is also feasible to carry out software reset for the whole chip or some modules.

   ✧ Setting address 0x6f[5] to 1b'1 is to reset the whole chip. Similar to watchdog reset (see **section 2.4.3 Power-saving mode**), retention analog registers afe_0x3a~ afe_0x3f are non-volatile, while other registers including afe_0x34~ afe_0x39 will be cleared by chip software reset.

   ✧ Addresses 0x60~0x62 serve to reset individual modules: if some bit is set to logic "1", the corresponding module is reset.

Table 2- 3 Register configuration for software reset

| Address | Mnemonic | Type | Description | Reset Value |
|---|---|---|---|---|
| 0x60 | RST0 | R/W | Reset control, 1 for reset, 0 for clear<br>[0]: SPI<br>[1]: I2C<br>[2]: n/a<br>[3]: n/a<br>[4]: MCU<br>[5]: n/a<br>[6]: AIF<br>[7]: ZB | 0xc0 |
| 0x61 | RST1 | R/W | [0]: system_timer<br>[1]: algm<br>[2]: dma<br>[3]: rs232<br>[4]: pwm<br>[5]: aes<br>[6]: n/a<br>[7]: swires | 0x3f |
| 0x62 | RST2 | R/W | [0]: n/a<br>[1]: n/a<br>[2]: n/a<br>[3]: adc<br>[4]: mcic<br>[5]: soft reset to reset mcic enable | 0x88 |

Shared under NDA

| Address | Mnemonic | Type | Description | Reset Value |
|---|---|---|---|---|
| | | | [6]: rsvd (mspi)<br>[7]: alg | |
| 0x6f | PWDNEN | W | [0]: suspend enable<br>[5]: rst all (act as watchdog reset)<br>[6]: rsvd (mcu low power mode)<br>[7]: stall mcu trig If bit[0] set 1, then system will go to suspend. Or only stall mcu. | 0x00 |

## 2.6 Power Management

The multiple-stage Power Management (PM) module is flexible to control power state of the whole chip or individual functional blocks such as MCU, RF Transceiver, and peripherals.

### 2.6.1 Digital LDO

The chip embeds LDO regulators to generate 1.8V regulated voltage. The internal LDO regulators serve to supply power for 1.8V digital core and analog modules in Active/Idle/Suspend mode.

While in deep sleep mode, the embedded 1.8V LDO regulators will be turned off.

### 2.6.2 VBUS LDO

The embedded VBUS LDO generates 3.0V voltage to supply power for the whole chip. The VBUS LDO supports two working modes, including LDO mode and Bypass mode.

- ✧ When the input voltage VBAT is 3.3V~4.3V, the VBUS LDO works in LDO mode.
- ✧ When the input voltage VBAT is lower than 3.3V, the VBUS LDO works in Bypass mode, and its output voltage is the same as VBAT input voltage.

### 2.6.3 Power-On-Reset (POR) and Brown-out detect

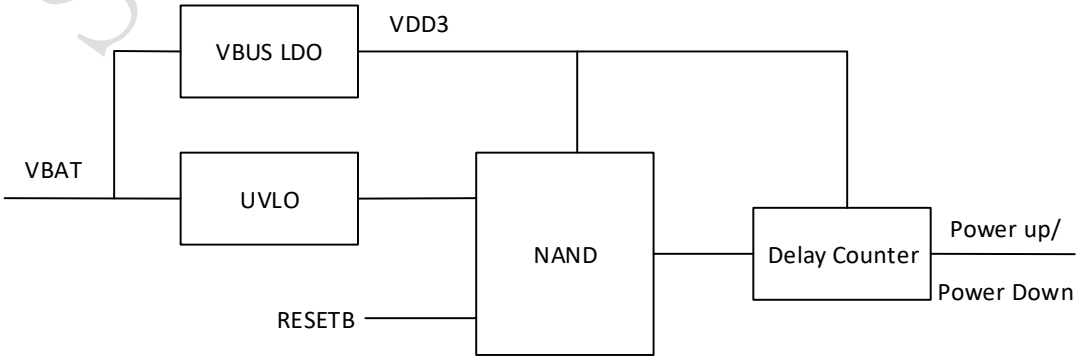



Figure 2- 4 Block diagram for power up/down

Shared under NDA

The whole chip power up and down is controlled by the UVLO (Ultra-low Voltage Lockout) module and the external RESETB pin via the logic shown in the above diagram. UVLO takes the external power supply as input and releases the lock only when the power supply voltage is higher than a preset threshold. The RESETB pin has an internal pull-up resistor; an external Cap can be connected on the RESETB pin to control the POR delay.

After both UVLO and RESETB release, there is further configurable **delay** before the system reset signal ("**Sysrst**") is released. This delay is adjusted by analog register afe_0x20. Since the content of afe_0x20 is reset to default only after power cycle, watchdog reset, or software reset, the delay change using afe_0x20 is only applicable when the chip has not gone through these reset conditions. For example, after deep sleep wakeup, the setting in afe_0x20 will take effect.

Table 2- 4 Analog register to control delay counter

| Address | Description | Default |
|---|---|---|
| afe_0x20 | r_dly:<br>[6:0]: delay, 32kHz decrease counter. Default delay 1ms.<br>[7] rsvd | 0xe0 |

***Note:** The register afe_0x20 will be reset to default after power cycle (POR), watchdog reset, or software reset.

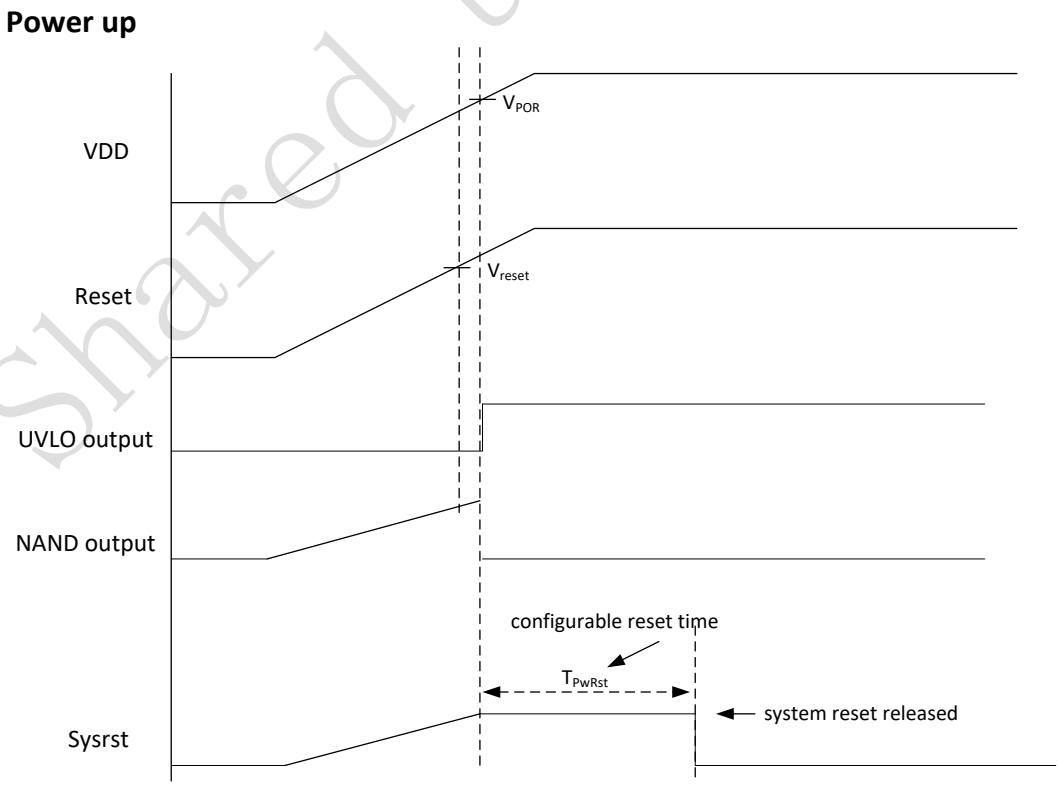



Figure 2- 5 Power-up sequence

Shared under NDA

## Power down

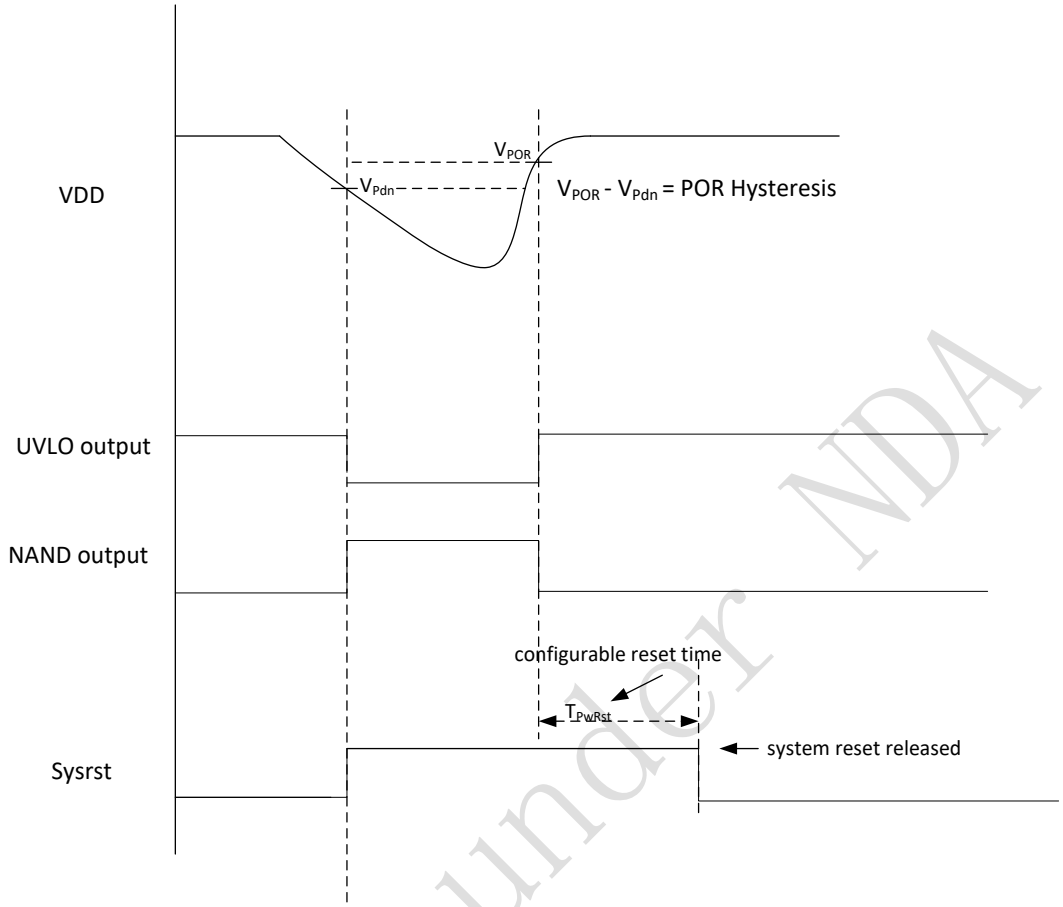



Figure 2- 6 Power-down sequence

The power up and power down sequence is shown in Table 14- 4, with the following parameters:

- $V_{POR}$ : VDD voltage when $V_{UVLO}$ turns to high level
- $V_{Pdn}$ : VDD voltage when $V_{UVLO}$ turns to low level
- $T_{PwRst}$ : Delay counter value (Configurable via analog register afe_0x20)

Shared under NDA

### 2.6.4 Working mode switch

The chip can switch to idle mode to stall the MCU.

To minimize power consumption, the chip can switch to power saving mode (suspend or deep sleep) correspondingly. In this case, the low-power 32kHz RC oscillator can still be running, and the low frequency wakeup timer LTIMER can be programmed to stay alive. The device can be activated to working state via external pin trigger or internal wakeup timer.

User can directly invoke corresponding library function to switch working mode of the chip.

If certain module doesn't need to work, user can power down this module in order to save power.

Table 2- 5 Analog registers for module power up/down control

| Address | Local name | Default Value | Description |
|---|---|---|---|
| afe_0x05<0> | 32K_rc_pd | 0 | Power down 32kHz RC oscillator<br>1: Power down<br>0: Power up |
| afe_0x05<2> | 24M_rc_pd | 0 | Power down of 24MHz RC oscillator<br>1: Power down<br>0: Power up |
| afe_0x05<3> | xtal_LDO_pd | 0 | Power down of 24MHz crystal oscillator<br>1: Power down<br>0: Power up |
| afe_0x05<4> | ldo_ana_pd | 0 | Power down of analog LDO<br>1: Power down<br>0: Power up |
| afe_0x06<1> | rx_lnaLDO_pd | 1 | Power down LNA LDO in RF transceiver<br>1: Power down<br>0: Power up |
| afe_0x06<2> | rx_anaLDO_pd | 1 | Power down analog LDO in RF transceiver<br>1: Power down<br>0: Power up |
| afe_0x06<3> | rx_rfLDO_pd | 1 | Power down RF LDO in RF transceiver<br>1: Power down<br>0: Power up |
| afe_0x06<6> | pll_vco_ldo_pd | 1 | Power down VCO LDO<br>1: Power down<br>0: Power up |

Shared under NDA

## 2.7 Wakeup sources

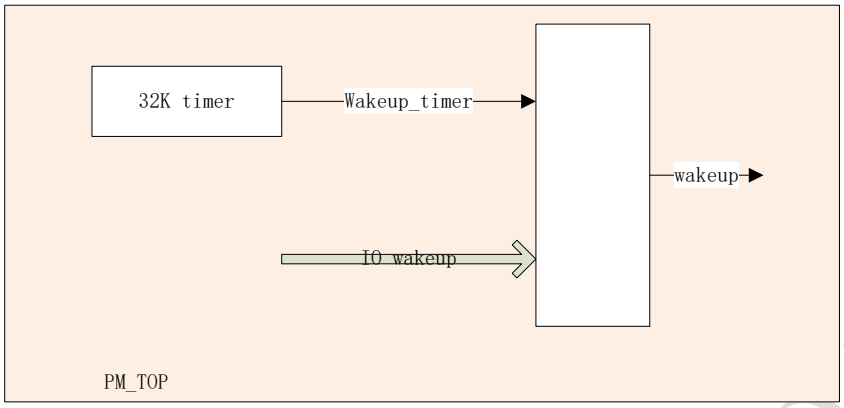



Figure 2- 7 Wakeup sources

### 2.7.1 Wakeup source – 32kHz timer

This wakeup source is able to wake up the system from suspend mode or deep sleep mode.

Address afe_0x26 bit[6] is the enabling bit for wakeup source from 32kHz timer.

### 2.7.2 Wakeup source – IO

This wakeup source is able to wake up the system from suspend mode or deep sleep mode. And IO wakeup supports high level or low level wakeup which is configurable via wakeup polarity control registers. Total wakeup pin can be up to 11.

Address afe_0x26[4] should be set as 1b'1 to enable IO wakeup source.

Enabling control registers: PA[7:5] enabling control register is afe_0x27[7:5], PB[7:0] enabling control register is afe_0x28[7:0].

Polarity control registers: PA[7:5] polarity control register is afe_0x21[7:5], PB[7:0] polarity control register is afe_0x22[7:0].

The corresponding driver is available so that user can directly invoke it to use IO wakeup source.

### 2.7.3 Register table

Table 2- 6Analog registers for Wakeup

| Address | R/W | Description | Default Value |
|---|---|---|---|
| afe_0x21 | R/W | pa_polarity:<br>[7:5]: select polarity for PA[7]~PA[5] IO (pad) wakeup<br>0: high level, 1: low level | 0x00 |
| afe_0x22 | R/W | pb_polarity:<br>[7:0]: select polarity for PB[7]~PB[0] IO (pad) wakeup<br>0: high level, 1: low level | 0x00 |
| afe_0x26[3] | R/W | Enable/Mask filter for IO (Pad) wakeup<br>1: Select 16us filter to filter out jitter on IO PAD input. | 0x00 |

Shared under NDA

| Address | R/W | Description | Default Value |
|---|---|---|---|
| | | 0: IO Pad combinational logic output (disable filter) | |
| afe_0x26[4] | R/W | 1: Enable IO (pad) wakeup | |
| afe_0x26[5] | R/W | Rsvd (Enable digital core wakeup) | |
| afe_0x26[6] | R/W | 1: Enable 32kHz timer wakeup | |
| afe_0x27 | R/W | pad_wkup_pa_en:<br>[7:5]: enable/disable PA[7]~PA[5] IO (pad) wakeup<br>1: enable; 0: disable | 0x00 |
| afe_0x28 | R/W | pad_wkup_pb_en:<br>[7:0]: enable/disable PB[7]~PB[0] IO (pad) wakeup<br>1: enable; 0: disable | 0x00 |
| afe_0x44 | W1C | State flag bits<br>[1]: pm_irq, i.e. 32kHz timer wakeup status<br>[2]: rsvd (digital core wakeup status)<br>[3] wkup_pad, i.e. IO (pad) wakeup status.<br>Write 1 to clean.<br>e.g. If bit[3] is 1, it indicates the system is waked up by IO (pad) source. | 0x00 |

Table 2- 7 Digital register for Wakeup

| Address | Mnemonic | Type | Description | Reset Value |
|---|---|---|---|---|
| 0x6e | WAKEUPEN | R/W | Wakeup enable<br>[0]: rsvd (enable wakeup from I2C host)<br>[1]: rsvd (enable wakeup from SPI host)<br>[2]: rsvd<br>[3]: enable wakeup from gpio<br>[4]: rsvd (enable wakeup from QDEC synchronous interface)<br>System resume control<br>[7]: sleep wakeup reset system enable | 1f |

Shared under NDA

# 3 2.4G RF Transceiver

## 3.1 Block diagram

The TLSR8367 integrates an advanced 2.4GHz RF transceiver. The RF transceiver works in the worldwide 2.4GHz ISM (Industrial Scientific Medical) band and contains an integrated Balun with a single-ended RF Tx/Rx port pin. No matching components are needed.

The transceiver consists of a fully integrated frequency synthesizer, a power amplifier, a modulator and a receiver. The transceiver can be configured to work in 2.4GHz proprietary 2Mbps/1Mbps/500kbps/250kbps mode.

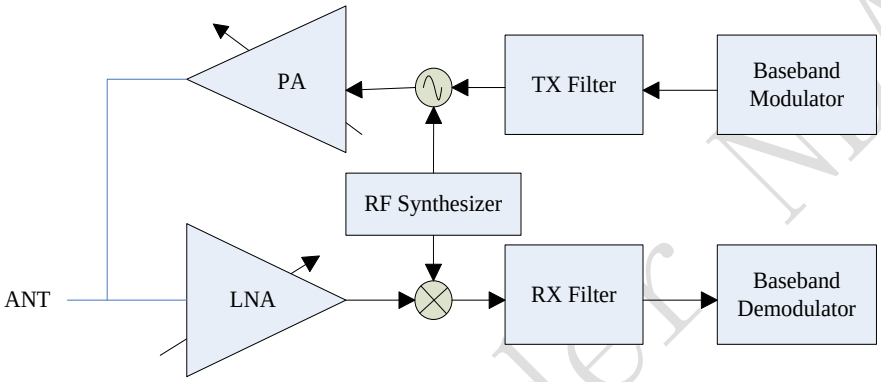



Figure 3- 1 Block diagram of RF transceiver

The internal PA can deliver up to 7dBm output power without the needs for an external RF PA. If higher output power is needed, then an external RF Frontend can be added to boost the output power.

To control **external** RF transceiver with both PA and LNA, first follow the GPIO lookup table (see section 7.1.1.1 GPIO lookup table) to configure the specific two pins as TX_CYC2PA and RX_CYC2LNA function, respectively (Note that GPIO function should be disabled at the same time). After the two pins are configured as TX_CYC2PA and RX_CYC2LNA function, the output function is enabled. Generally the two pins are both high active.

Table 3- 1 External RF transceiver control example

| TX_CYC2PA | RX_CYC2LNA | External RF transceiver |
|---|---|---|
| L | L | Both LNA and PA OFF |
| L | H | LNA ON |
| H | L | PA ON |
| H | H | N/A |

Shared under NDA

## 3.2 Function description

### 3.2.1 Air interface data rate and RF channel frequency

Air interface data rate, the modulated signaling rate for RF transceiver when transmitting and receiving data, is configurable via related register setting: 250kbps, 500kbps, 1Mbps, 2Mbps.

For the TLSR8367, RF transceiver can operate with frequency ranging from 2.400GHz to 2.4835GHz. The RF channel frequency setting determines the center of the channel.

## 3.3 Baseband

The baseband is disabled by default. The corresponding API is available for user to power on/down the baseband and enable/disable clock, so that the baseband can be turned on/off flexibly.

The baseband contains dedicated hardware logic to perform fast AGC control, access code correlation, CRC checking, data whitening, encryption/decryption and frequency hopping logic.

The baseband supports all features required by 2.4GHz proprietary specification.

### 3.3.1 Packet format

Packet format in 2.4GHz Proprietary mode is shown as Table 3- 2:

Table 3- 2 Packet format in Proprietary mode

LSB MSB

| Preamble (8 bits) | Address code (configurable 3~5 bytes) | Packet Controller + Payload (1~63 bytes) | CRC (1~2 bytes) |
|---|---|---|---|

### 3.3.2 RSSI and frequency offset

The TLSR8367 provides accurate RSSI (Receiver Signal Strength Indicator) and frequency offset indication.

- ✧ 1Byte RSSI can be read upon receiving the data packet.
- ✧ If no data packet is received (e.g. to perform channel energy measurement when no desired signal is present), real-time RSSI can also be read from specific registers which will be updated automatically.
- ✧ RSSI resolution can reach +/-4dB.
- ✧ 2-Byte Frequency offset value can be read upon receiving the data packet. Valid bits of actual frequency offset may be less than 16bits, and different valid bits correspond to different tolerance range.

Telink supplies corresponding drivers for user to read RSSI and frequency offset as needed.

Shared under NDA

# 4 Clock

## 4.1 Clock sources

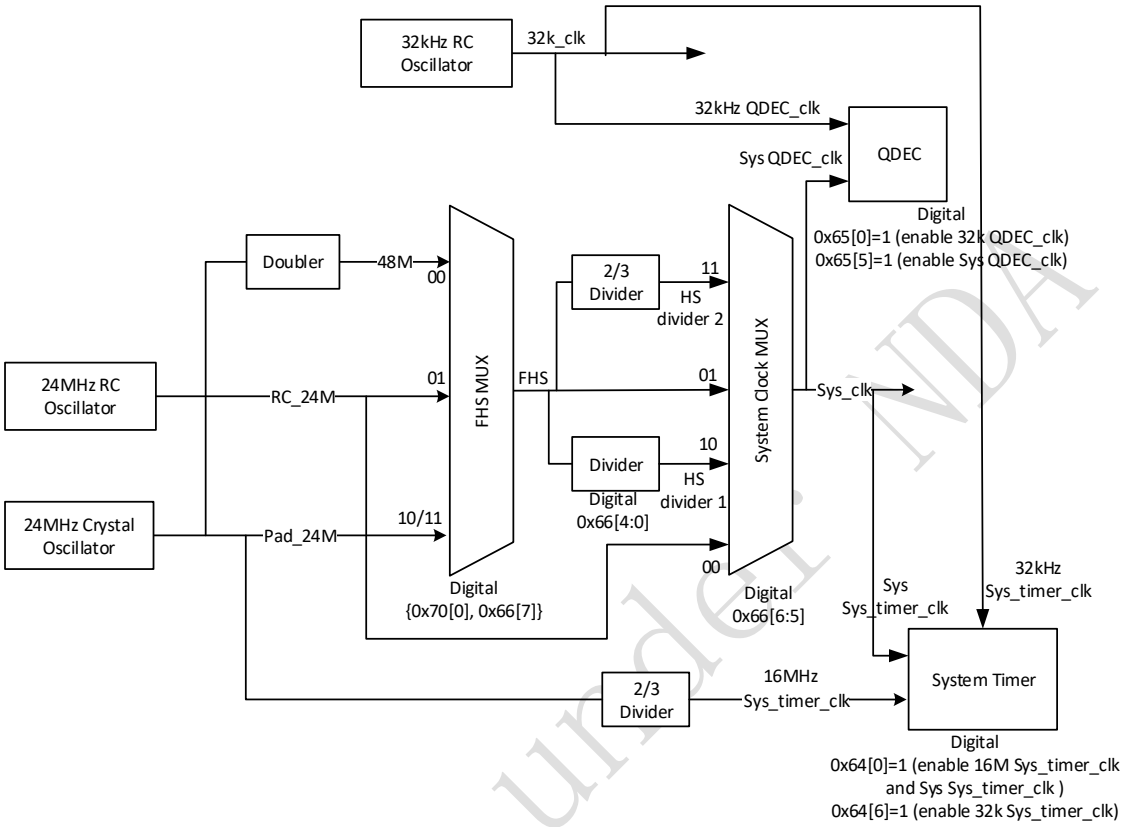



Figure 4- 1 Block diagram of clock

The TLSR8367 clock supports 24MHz external crystal or 24MHz/32kHz embedded RC oscillator.

- A **RC_24M** clock is available from an embedded 24MHz RC oscillator. It can be directly used as clock source for system, and it's also a selectable source for high speed clock (FHS). When system clock is configured as 24MHz RC (default), the RC_24M clock can be output to PA[7] by configuring the digital registers as below: 0x586=0x7f, 0x5a9=0x40.
- A **RC_32k** clock is available from an embedded 32kHz RC oscillator. It can provide a 32kHz clock source for 32kHz timer during sleep state as well as System Timer and QDEC module. When 32kHz clock is configured as 32kHz RC (default), the RC_32k clock can be output to PA[1] by configuring the digital registers as below: 0x586=0xfd, 0x5a8=0x08.
- A **Pad_24M** clock is available from external 24MHz crystal via pin XC1 and XC2. It can provide 16MHz clock source for System Timer via a 2/3 frequency divider. The Pad_24M can be directly used as clock source for high speed clock (FHS), or generate a 48MHz clock source via a frequency doubler for FHS.
- High speed clock (FHS) can be directly used as clock source for system, or generate HS divider clock 1 or 2 source via a configurable or fixed 2/3 frequency divider for system.

Shared under NDA

## 4.2 System clock

There are four selectable clock sources for MCU system clock: **RC_24M** (derived from 24MHz RC oscillator), High speed clock "**FHS**", **HS divider clock 1** (derived from "FHS" via a configurable frequency divider), **HS divider clock 2** (derived from "FHS" via a fixed 2/3 frequency divider).

The high speed clock (FHS) is selectable via address {0x70[0], 0x66[7]} from the following sources: **48MHz** clock (derived from 24MHz crystal oscillator via a frequency doubler), **RC_24M** (derived from 24MHz RC oscillator), and **Pad_24M** (derived from 24MHz crystal oscillator).

The digital register CLKSEL (address 0x66) serves to set system clock: System clock source is selectable via bit[6:5].

- ✧ If address 0x66[6:5] is set to 2b'00 to select the RC_24M, system clock frequency equals 24MHz.
- ✧ If address 0x66[6:5] is set to 2b'01 to select the FHS clock, system clock frequency equals the FHS frequency ($F_{FHS}$).
- ✧ If address 0x66[6:5] is set to 2b'10 to select the HS divider clock 1, system clock frequency is adjustable via address 0x66[4:0]. The formula is shown as below:

  $F_{System\ clock} = F_{FHS}$ / (system clock divider value in address 0x66[4:0]).
- ✧ If address 0x66[6:5] is set to 2b'11 to select the HS divider clock 2, system clock frequency equals $F_{FHS}$ * 2/3.

## 4.3 Module clock

Registers CLKEN0~CLKEN2 (address 0x63~0x65) are used to enable or disable clock for various modules. By disable the clocks of unused modules, current consumption could be reduced.

### 4.3.1 System Timer clock

System Timer simultaneously uses system clock, a 16MHz clock, as well as a 32kHz clock.

- ✧ The 16MHz clock is derived from 24MHz crystal oscillator via a 2/3 frequency divider.
- ✧ The 32kHz clock is **RC_32k**.
- ✧ Digital register 0x64 bit[0] and bit[6] should be enabled to drive the System Timer by the 16MHz clock, system clock, as well as the 32kHz clock.

### 4.3.2 QDEC clock

QDEC module simultaneously uses system clock as well as a 32kHz clock.

- ✧ The 32kHz clock is **RC_32k**.
- ✧ Digital register 0x65 bit[0] and bit[5] should be enabled to drive the whole QDEC module by the 32kHz clock and system clock.

Shared under NDA

## 4.4 Register table

Table 4- 1Register table related to clock

| Address | Mnemonic | R/W | Description | Default |
|---|---|---|---|---|
| **Digital register** | | | | |
| 0x63 | CLKEN0 | R/W | Clock enable control: 1 for enable; 0 for disable<br>[0]: SPI<br>[1]: I2C<br>[2]: HOSTIRQ<br>[3]: n/a<br>[4]: MCU<br>[5]: FPU (Float Point Unit)<br>[6]: AIF<br>[7]: ZB | 13 |
| 0x64 | CLKEN1 | R/W | Clock enable control: 1 for enable; 0 for disable<br>[0]: System timer (16M_clk and sys_clk for System timer)<br>[1]: ALGM<br>[2]: DMA<br>[3]: RS232<br>[4]: PWM<br>[5]: AES<br>[6]: 32k_clk for system timer<br>[7]: Swires | 80 |
| 0x65 | CLKEN2 | R/W | Clock enable control: 1 for enable; 0 for disable<br>[0]: 32k_clk for QDEC<br>[1]: 32k_clk for PWM<br>[2:3]: n/a<br>[4]: MCIC<br>[5]: QDEC (sys_clk for QDEC)<br>[6:7]: n/a | 10 |
| 0x66 | CLKSEL | R/W | System clock select<br>[4:0]: system clock divider.<br>If 0x66[6:5] is set as 2b' 10, $F_{Sysclk} = F_{FHS}$ / CLKSEL[4:0].<br>FHS: refer to 0x70 FHS_sel.<br>[6:5]: select system clock source<br>2'b00: RC_24M from RC oscillator<br>2'b01: FHS clock (High speed clock)<br>2'b10: HS divider clock 1 derived from FHS clock via a configurable divider (see 0x66[4:0])<br>2'b11: HS divider clock 2 derived from FHS clock via a fixed 2/3 divider<br>{0x70[0], 0x66[7]}: FHS select | 06 |
| 0x70 | FHS_sel | R/W | {0x70[0], 0x66[7]}: FHS select<br>2'b00: 48MHz clock doubled from 24MHz crystal oscillator<br>2'b01: RC_24M from RC oscillator<br>2'b1x: Pad_24M from 24MHz crystal oscillator | 00 |

Shared under NDA

# 5 Timers

## 5.1 Timer0~Timer2

The TLSR8367 supports three general 32-bit timers in active mode, including Timer0~ Timer2. All of the three timers support four modes: Mode 0 (System Clock Mode), Mode 1 (GPIO Trigger Mode), Mode 2 (GPIO Pulse Width Mode) and Mode 3 (Tick Mode), which are selectable via the register TMR_CTRL0 (address 0x620) ~ TMR_CTRL1 (address 0x621).

Timer 2 can also be configured as “watchdog” timer to monitor firmware running.

### 5.1.1 Register table

Table 5- 1 Register configuration for Timer0~Timer2

| Address | Mnemonic | Type | Description | Reset Value |
|---|---|---|---|---|
| 0x72 | Wd_status | R/W | [0] watch dog status, write 1 to clear. | 00 |
| 0x620 | TMR_CTRL0 | RW | [0]Timer0 enable<br>1: enable<br>[2:1] Timer0 mode.<br>0: using sclk, 1: using gpio,<br>2: count width of gpi, 3: tick<br>[3]Timer1 enable<br>[5:4] Timer1 mode.<br>[6]Timer2 enable<br>[7]Bit of timer2 mode | 00 |
| 0x621 | TMR_CTRL1 | RW | [0]Bit of timer2 mode<br>[7:1]Low bits of watch dog capture | 00 |
| 0x622 | TMR_CTRL2 | RW | [6:0]High bits of watch dog capture. Watch dog capture is compared with [31:18] of timer2 ticker.<br>[7]watch dog capture enable<br>1: enable | 00 |
| 0x623 | TMR_STATUS | RW | [0] timer0 status, write 1 to clear<br>[1] timer1 status, write 1 to clear<br>[2] timer2 status, write 1 to clear | 00 |
| 0x624 | TMR_CAPT0_0 | RW | Byte 0 of timer0 capture | 00 |
| 0x625 | TMR_CAPT0_1 | RW | Byte 1 of timer0 capture | 00 |
| 0x626 | TMR_CAPT0_2 | RW | Byte 2 of timer0 capture | 00 |
| 0x627 | TMR_CAPT0_3 | RW | Byte 3 of timer0 capture | 00 |
| 0x628 | TMR_CAPT1_0 | RW | Byte 0 of timer1 capture | 00 |
| 0x629 | TMR_CAPT1_1 | RW | Byte 1 of timer1 capture | 00 |
| 0x62a | TMR_CAPT1_2 | RW | Byte 2 of timer1 capture | 00 |
| 0x62b | TMR_CAPT1_3 | RW | Byte 3 of timer1 capture | 00 |
| 0x62c | TMR_CAPT2_0 | RW | Byte 0 of timer2 capture | 00 |
| 0x62d | TMR_CAPT2_1 | RW | Byte 1 of timer2 capture | 00 |

Shared under NDA

| Address | Mnemonic | Type | Description | Reset Value |
|---|---|---|---|---|
| 0x62e | TMR_CAPT2_2 | RW | Byte 2 of timer2 capture | 00 |
| 0x62f | TMR_CAPT2_3 | RW | Byte 3 of timer2 capture | 00 |
| 0x630 | TMR_TICK0_0 | RW | Byte 0 of timer0 ticker | 00 |
| 0x631 | TMR_TICK0_1 | RW | Byte 1 of timer0 ticker | 00 |
| 0x632 | TMR_TICK0_2 | RW | Byte 2 of timer0 ticker | 00 |
| 0x633 | TMR_TICK0_3 | RW | Byte 3 of timer0 ticker | 00 |
| 0x634 | TMR_TICK1_0 | RW | Byte 0 of timer1 ticker | 00 |
| 0x635 | TMR_TICK1_1 | RW | Byte 1 of timer1 ticker | 00 |
| 0x636 | TMR_TICK1_2 | RW | Byte 2 of timer1 ticker | 00 |
| 0x637 | TMR_TICK1_3 | RW | Byte 3 of timer1 ticker | 00 |
| 0x638 | TMR_TICK2_0 | RW | Byte 0 of timer2 ticker | 00 |
| 0x639 | TMR_TICK2_1 | RW | Byte 1 of timer2 ticker | 00 |
| 0x63a | TMR_TICK2_2 | RW | Byte 2 of timer2 ticker | 00 |
| 0x63b | TMR_TICK2_3 | RW | Byte 3 of timer2 ticker | 00 |

### 5.1.2 Mode0 (System Clock Mode)

In Mode 0, system clock is used as clock source.

After Timer is enabled, Timer Tick (i.e. counting value) is increased by 1 on each positive edge of system clock from preset initial Tick value. Generally the initial Tick value is set as 0.

Once current Timer Tick value matches the preset Timer Capture (i.e. timing value), Timer stops counting, Timer status is updated, and an interrupt is generated (if enabled).

Following is an example to show steps of setting Timer0 as Mode 0.

**1st: Set initial Tick value of Timer0**

Set initial Tick value of Timer0 via registers TMR_TICK0_0~TMR_TICK0_3 (address 0x630~0x633). Address 0x630 is lowest byte and 0x633 is highest byte. It's recommended to clear initial Timer Tick value to 0.

**2nd: Set Capture value of Timer0**

Set registers TMR_CAPT0_0~TMR_CAPT0_3 (address 0x624~0x627). Address 0x624 is lowest byte and 0x627 is highest byte.

**3rd: Set Timer0 as Mode 0 and enable Timer0**

Set register TMR_CTRL0 (address 0x620) [2:1] as 2b'00 to select Mode 0; Meanwhile set address 0x620[0] as 1b'1 to enable Timer0. Timer0 starts counting upward, and Tick value is increased by 1 on each positive edge of system clock until it reaches Timer0 Capture value.

### 5.1.3 Mode1 (GPIO Trigger Mode)

In Mode 1, GPIO is used as clock source. The "**m0**"/"**m1**"/"**m2**" register specifies the GPIO which generates counting signal for Timer0/Timer1/Timer2.

Shared under NDA

After Timer is enabled, Timer Tick (i.e. counting value) is increased by 1 on each positive/negative (configurable) edge of GPIO from preset initial Tick value. Generally the initial Tick value is set as 0. The "**Polarity**" register specifies the GPIO edge when Timer Tick counting increases.

**Note**: Refer to **Section 7.1.2** for corresponding "**m0**", "**m1**", "**m2**" and "**Polarity**" register address.

Once current Timer Tick value matches the preset Timer Capture (i.e. timing value), timer stops counting and an interrupt is generated (if enabled).

Following is an example to show steps of setting Timer1 as Mode 1.

**1st: Set initial Tick value of Timer1**

Set initial Tick value of Timer1 via registers TMR_TICK1_0~TMR_TICK1_3 (address 0x634~0x637). Address 0x634 is lowest byte and 0x637 is highest byte. It's recommended to clear initial Timer Tick value to 0.

**2nd: Set Capture value of Timer1**

Set registers TMR_CAPT1_0~TMR_CAPT1_3 (address 0x628~0x62b). Address 0x628 is lowest byte and 0x62b is highest byte.

**3rd: Select GPIO source and edge for Timer1**

Select certain GPIO to be the clock source via setting "m1" register.

Select positive edge or negative edge of GPIO input to trigger Timer1 Tick increment via setting "Polarity" register.

**4th: Set Timer1 as Mode 1 and enable Timer1**

Set address 0x620[5:4] as 2b'01 to select Mode 1; Meanwhile set address 0x620[3] as 1b'1 to enable Timer1. Timer1 starts counting upward, and Timer1 Tick value is increased by 1 on each positive/negative (specified during the 3rd step) edge of the specified GPIO input until it reaches Timer1 Capture value.

### 5.1.4 Mode2 (GPIO Pulse Width Mode)

In Mode 2, system clock is used as the unit to measure the width of GPIO pulse. The "**m0**"/"**m1**"/"**m2**" register specifies the GPIO which generates control signal for Timer0/Timer1/Timer2.

After Timer is enabled, Timer Tick is triggered by a positive/negative (configurable) edge of GPIO pulse. Then Timer Tick (i.e. counting value) is increased by 1 on each positive edge of system clock from preset initial Tick value. Generally the initial Tick value is set as 0. The "**Polarity**" register specifies the GPIO edge when Timer Tick starts counting.

**Note**: Refer to **Section 7.1.2** for corresponding "**m0**", "**m1**", "**m2**" and "**Polarity**" register address.

While a negative/positive edge of GPIO pulse is detected, timer stops counting and an interrupt is generated (if enabled). The GPIO pulse width could be calculated in terms of tick count and period of system clock.

Following is an example to show steps of setting Timer2 as Mode 2.

**1st: Set initial Timer2 Tick value**

Set Initial value of Tick via registers TMR_TICK2_0~TMR_TICK2_3 (address 0x638~0x63b). Address 0x638 is lowest byte and 0x63b is highest byte. It's recommended to clear initial Timer Tick

Shared under NDA

value to 0.

**2nd: Select GPIO source and edge for Timer2**

Select certain GPIO source via setting “m2” register.

Select positive edge or negative edge of GPIO input to trigger Timer2 counting start via setting “Polarity” register.

**3rd: Set Timer2 as Mode 2 and enable Timer2**

Set address 0x620[7:6] to 2b’01 and address 0x621 [0] to 1b’1.

Timer2 Tick is triggered by a positive/negative (specified during the 2nd step) edge of the specified GPIO pulse. Timer2 starts counting upward and Timer2 Tick value is increased by 1 on each positive edge of system clock.

While a negative/positive edge of GPIO pulse is detected, Timer2 tick stops and an interrupt is generated (if enabled).

**4th: Read current Timer2 Tick value to calculate GPIO pulse width**

Read current Timer2 Tick value from address 0x638~0x63b.

Then GPIO pulse width is calculated as follows:

$$\text{GPIO pulse width} = \text{System clock period} * (\text{current Timer2 Tick} - \text{intial Timer2 Tick})$$

For initial Timer2 Tick value is set to the recommended value of 0, then:

$$\text{GPIO pulse width} = \text{System clock period} * \text{current Timer2 Tick}.$$

### 5.1.5 Mode3 (Tick Mode)

In Mode 3, system clock is used as clock source.

After Timer is enabled, Timer Tick starts counting upward, and Timer Tick value is increased by 1 on each positive edge of system clock.

This mode could be used as time indicator. No interrupt will be generated. Timer Tick keeps rolling from 0 to 0xffffffff. When Timer tick overflows, it returns to 0 and starts counting upward again.

Following is an example to show steps of setting Timer0 as Mode 3.

**1st: Set initial Tick value of Timer0**

Set Initial value of Tick via address 0x630~0x633. Address 0x630 is lowest byte and address 0x633 is highest byte. It’s recommended to clear initial Timer Tick value to 0.

**2nd: Set Timer0 as Mode 3 and enable Timer0**

Set address 0x620[2:1] as 2b’11 to select Mode 3, meanwhile set address 0x620[0] as 1b’1 to enable Timer0. Timer0 Tick starts to roll.

**3rd: Read current Timer0 Tick value**

Current Timer0 Tick value can be read from address 0x630~0x633.

Shared under NDA

### 5.1.6 Watchdog

Only Timer2 can be used as a watchdog timer, so that it could reset chip from unexpected hang up or malfunction.

Timer2 Tick has 32bits. Watchdog Capture has only 14bits, which consists of TMR_CTRL2 (address 0x622) [6:0] as higher bits and TMR_CTRL1 (address 0x621) [7:1] as lower bits. Chip will be reset when the Timer2 Tick[31:18] matches Watchdog Capture value.

The range of duration that can be set in watchdog is between T(sysclk)*2^18 and T(sysclk)*2^32, and T(sysclk) is the system clock period which is configurable. For example, if system clock is configured to 16MHz, then the upper time limit of watchdog reset is 2^32/16MHz ≈ 268s.

Following shows steps of setting Timer2 as watchdog timer.

**1st: Clear Timer2 Tick value**

Clear registers TMR_TICK2_0 ~TMR_TICK2_3 (address 0x638~0x63b). Address 0x638 is lowest byte and 0x63b is highest byte.

**2nd: Enable Timer2**

Set register TMR_CTRL0 (address 0x620) [6] as 1b'1 to enable Timer2.

**3rd: Set 14-bit Watchdog Capture value and enable Watchdog**

Set higher 7 bits and lower 7 bits of Watchdog Capture via address 0x622[6:0] and 0x621[7:1]. Meanwhile set address 0x622[7] as 1b'1 to enable Watchdog.

Then Timer2 Tick starts counting upwards from 0.

If bits[31:18] of Timer2 Tick value read from address 0x638~0x63b reaches Watchdog Capture, the chip will be reset, and the status bit in address 0x72[0] will be set as 1b'1 automatically. User can read the watchdog status bit after chip reset to check if the reset source is watchdog, and needs to write 1b'1 to this bit to manually clear the flag.

## 5.2 32kHz LTIMER

The TLSR8367 also supports a low frequency (32kHz) timer "LTIMER" in suspend mode or deep sleep mode. This 32kHz timer can be used as one kind of wakeup source.

32kHz LTIMER related functions are all handled in the stack.

## 5.3 System Timer

The TLSR8367 also supports a System Timer. Please refer to section **4.3.1** for System Timer clock.

In suspend mode, both System Timer and Timer0~Timer2 stop counting, and 32kHz Timer starts counting. When the chip restores to active mode, Timer0~Timer2 will continue counting from the number when they stops; In contrast, System Timer will continue counting from an adjusted number which is a sum of the number when it stops and an offset calculated from the counting value of 32kHz Timer during suspend mode.

System timer related functions are all handled in the stack.

Table 5- 2 Register table for System Timer

Shared under NDA

| Address | Mnemonic | R/W | Function | Default Value |
|---|---|---|---|---|
| 0x740 | Sys_timer[7:0] | R/W | | 0x00 |
| 0x741 | Sys_timer[15:8] | R/W | | 0x00 |
| 0x742 | Sys_timer[23:16] | R/W | | 0x00 |
| 0x743 | Sys_timer[31:24] | R/W | System timer counter, write to set initial value. | 0x00 |
| 0x748 | 32K_cal_latch[7:0] | R | 32k calibration count[7:0] | 0x00 |
| 0x749 | 32K_cal_latch[15:8] | R | 32k calibration count[15:8] | 0x00 |
| 0x74a | Sys_timer_ctrl | R/W | [7]: enable of system timer<br>[6]: irq_mask for system timer<br>1: enable, 0: disable<br>[5:4]: calibration mode<br>2'b00: 4 cycles of 32kHz clock<br>2'b01: 8 cycles of 32kHz clock<br>2'b10: 16 cycles of 32kHz clock (Default setting)<br>2'b11: 32 cycles of 32kHz clock<br>[3]: calibration enable<br>[2]: set to 0<br>[1]: rsvd<br>[0]: set 32kHz timer<br>1: write; 0: read | 0x21 |
| 0x74b | Sys_timer_cmd | WO | [7:6]: rsvd<br>[5]: clear 32k read latch update flag<br>[4]: rsvd<br>[3]: start 32k count write/read<br>[2:0]: rsvd | 0x00 |
| 0x74b | Sys_timer_status | RO | [7]: rsvd<br>[6]: rd_busy_man_2d<br>[5]: rd_update_man_2d<br>[4]: rsvd<br>[3]: ss_sync<br>[2]: cmd_set_tgl<br>[1:0]: rsvd | 0x00 |
| 0x74c | sys_timer_32K_set [7:0] | R/W | 32k timer write[7:0] | 0x00 |
| 0x74d | sys_timer_32K_set [15:8] | R/W | 32k timer write[15:8] | 0x00 |
| 0x74e | sys_timer_32K_set [23:16] | R/W | 32k timer write[23:16] | 0x00 |
| 0x74f | sys_timer_32K_set [31:24] | R/W | 32k timer write[31:24] | 0x00 |
| 0x750 | sys_timer_32K_read [7:0] | R | 32k timer read[7:0] | 0x00 |

Shared under NDA

| Address | Mnemonic | R/W | Function | Default Value |
|---|---|---|---|---|
| 0x751 | sys_timer_32K_read [15:8] | R | 32k timer read[15:8] | 0x00 |
| 0x752 | sys_timer_32K_read [23:16] | R | 32k timer read[23:16] | 0x00 |
| 0x753 | sys_timer_32K_read [31:24] | R | 32k timer read[31:24] | 0x00 |

***Note:**

32kHz clock calibration is related to 32k cycle numbers used. More cycles correspond to higher accuracy but more time. Generally 32kHz clock calibration will select 16 cycles of 32kHz clock.

The lower three bits of address 0x740 is invalid, therefore, the resolution should be 0.5us. The 0.5us resolution is only used for system timer read out. 32kHz clock calibration is done using every 16MHz clock cycle, so 32kHz clock accuracy is not affected.

1. Get 32kHz Timer count value

0x74a[0] = 0; //set to 32kHz Timer read mode

0x74a[3] = 1; //enable calibration to provide 16MHz clock count value related to 32kHz cycle (0x74a[5:4])

0x74a[7] = 1; //kick system timer to tick

0x74b[5] = 0; //clear 32kHz read update flag

Wait for 0x74b[5]==1; //wait for the next 32kHz update flag

Read 32kHz Timer value from 0x750

2. Set 32kHz Timer count value

0x74a[0] = 1; //set to 32kHz Timer write mode

Wait for 0x74b[6]==0; //see whether during 32kHz read

Write 32kHz Timer value to 0x74c;

0x74b[3] = 1; //start 32kHz write sync process

Wait for 0x74b[3] from 0 to 1; //wait 32kHz sync indicator from 16MHz to sys domain

Wait for 0x74b[3] from 1 to 0; //wait 32kHz sync indicator done

Shared under NDA

# 6 Interrupt System

## 6.1 Interrupt structure

The interrupting function is applied to manage dynamic program sequencing based on real-time events triggered by timers, pins and etc.

For the TLSR8367, there are 24 interrupt sources in all: 16 types are level-triggered interrupt sources (listed in address 0x640~0x641) and 8 types are edge-triggered interrupt sources (listed in address 0x642).

When CPU receives an interrupt request (IRQ) from some interrupt source, it will determine whether to respond to the IRQ. If CPU decides to respond, it pauses current routine and starts to execute interrupt service subroutine. Program will jump to certain code address and execute IRQ commands. After finishing interrupt service subroutine, CPU returns to the breakpoint and continues to execute main function.

## 6.2 Register configuration

Table 6- 1 Register table for Interrupt system

| Address | Mnemonic | Type | Description | Reset Value |
|---|---|---|---|---|
| 0x640 | MASK_0 | RW | Byte 0 interrupt mask, level-triggered type<br>{irq_host_cmd, irq_uart, rsvd, irq_dma, rsvd, time2, time1, time0}<br>[7] irq_host_cmd<br>[6] irq_uart<br>[5] rsvd<br>[4] irq_dma<br>[3] rsvd<br>[2] time2<br>[1] time1<br>[0] time0 | 0x00 |
| 0x641 | MASK_1 | RW | Byte 1 interrupt mask, level-triggered type<br>{an_irq, irq_pwm, irq_zb_rt, irq_software, 4'b0}<br>[7] an_irq<br>[6] irq_pwm<br>[5] irq_zb_rt<br>[4] irq_software<br>[3:0] rsvd | 0x00 |
| 0x642 | MASK_2 | RW | Byte 2 interrupt mask, edge-triggered type<br>{gpio2risc[2:0], 1'b0, pm_irq, irq_gpio, 2'b0}<br>[7] gpio2risc[2]<br>[6] gpio2risc[1]<br>[5] gpio2risc[0]<br>[4] irq_stimer, used together with address 0x74a[6]<br>[3] pm_irq<br>[2] irq_gpio<br>[1:0] rsvd | 0x00 |

Shared Under NDA

| Address | Mnemonic | Type | Description | Reset Value |
|---|---|---|---|---|
| 0x643 | IRQMODE | RW | [0] interrupt enable<br>[1] reserved (Multi-Address enable) | 0x00 |
| 0x644 | PRIO_0 | RW | Byte 0 of priority<br>1: High priority; 0: Low priority | 0x00 |
| 0x645 | PRIO_1 | RW | Byte 1 of priority | 0x00 |
| 0x646 | PRIO_2 | RW | Byte 2 of priority | 0x00 |
| 0x648 | IRQSRC_0 | R | Byte 0 of interrupt source | 0x00 |
| 0x649 | IRQSRC_1 | R | Byte 1 of interrupt source | 0x00 |
| 0x64a | IRQSRC_2 | R | Byte 2 of interrupt source | 0x00 |

### 6.2.1 Enable/Mask interrupt sources

Various interrupt sources could be enabled or masked by registers MASK_0~MASK_2 (address 0x640~0x642).

Interrupt sources of level-triggered type:

- ✧ irq_host_cmd (0x640[7]): I2C Slave mapping mode or SPI Slave interrupt
- ✧ irq_uart (0x640[6]): UART interrupt
- ✧ irq_dma (0x640[4]): DMA interrupt
- ✧ time2, time1, timer0 (0x640[2]~0x640[0]): Timer2~Timer0 interrupt
- ✧ an_irq (0x641[7]): pm_irq, gpio2risc[2], gpio2risc[1] or gpio2risc[0] interrupt selectable via digital register 0x26[2:0], not recommended to use.
- ✧ irq_pwm (0x641[6]): PWM interrupt
- ✧ irq_zb_rt (0x641[5]): Baseband interrupt
- ✧ irq_software (0x641[4]): Software interrupt

Interrupt sources of edge-triggered type:

- ✧ gpio2risc[2:0] (0x642[7]~0x642[5]): gpio2risc[2]~gpio2risc[0] interrupt, please refer to section **7.1.2**.
- ✧ irq_stimer (0x642[4]): System timer interrupt, should be used together with address 0x74a[6]
- ✧ pm_irq (0x642[3]): 32kHz timer wakeup interrupt
- ✧ irq_gpio (0x642[2]): GPIO interrupt, please refer to section **7.1.2**.

### 6.2.2 Interrupt mode and priority

Interrupt mode is typically-used mode. Register IRQMODE (address 0x643)[0] should be set as 1b'1 to enable interrupt function.

IRQ tasks could be set as High or Low priority via registers PRIO_0~PRIO_2 (address 0x644~0x646). When more than one interrupt sources assert interrupt requests at the same time, CPU will respond depending on respective interrupt priority levels. It's recommended not to modify priority setting.

Shared under NDA

### 6.2.3 Interrupt source flag

Three bytes in registers IRQSRC_0~IRQSRC_2 (address 0x648~0x64a) serve to indicate IRQ sources. Once IRQ occurs from certain source, the corresponding IRQ source flag will be set as “1”. User could identify IRQ source by reading address 0x648~0x64a.

When handling edge-triggered type interrupt, the corresponding IRQ source flag needs to be cleared via address 0x64a. Take the interrupt source irq_gpio for example: First enable the interrupt source by setting address 0x642 bit[2] as 1b’1; then set address 0x643 bit[0] as 1b’1 to enable the interrupt. In interrupt handling function, 24-bit data is read from address 0x648~0x64a to check which IRQ source is valid; if data bit[18] is 1, it means the irq_gpio IRQ source is valid. Clear this interrupt source by setting address 0x64a bit[2] as 1b’1.

As for level-type interrupt, IRQ interrupt source status needs to be cleared via setting corresponding module status register. Take Timer0 IRQ interrupt source for example: First enable the interrupt source by setting address 0x640 bit[0] as 1b’1; then set address 0x643 bit[0] as 1b’1 to enable the interrupt. In interrupt handling function, 24-bit data is read from address 0x648~0x64a to check which IRQ source is valid; if data bit[0] is 1, it means the Timer0 IRQ source is valid. Register TMR_STATUS (address 0x623) [0] should be written with 1b’1 to manually clear Timer0 status (refer to section **5.1.1 Register table**).

Shared under NDA

# 7 Interface

## 7.1 GPIO

The TLSR8367EP16 support up to 9 GPIOs respectively. All digital IOs including PA[0]~PD[3] and can be used as GPIOs (general purpose IOs).

### 7.1.1 Basic configuration

Please refer to the table in **section 7.1.1.1** for various GPIO interface configuration.

#### 7.1.1.1 GPIO lookup table

Table 7- 1GPIO lookup table

| Pin | Default function | Pad Function Mux | | | | | GPIO Setting | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Register=3 | Register=2 | Register=1 | Register=0 | Register | Input (R) | IE | OEN | Output | Polarity | DS | Act as GPIO |
| PWM3/ PWM0/ RX_CYC2LNA/ PA[0] | GPIO | / | RX_CYC2LNA | PWM0 | PWM3 | 0x5a8[1:0] | 0x580[0] | afe_0xb6[0] | 0x582[0] | 0x583[0] | 0x584[0] | afe_0xb8[0] | 0x586[0] |
| PWM3_N/ PA[1] | GPIO | Rsvd | Rsvd | / | PWM3_N | 0x5a8[3:2] | 0x580[1] | afe_0xb6[1] | 0x582[1] | 0x583[1] | 0x584[1] | afe_0xb8[1] | 0x586[1] |
| PWM1_N/ pga_in<0>/PA[2] | GPIO | Rsvd | Rsvd | / | PWM1_N | 0x5a8[5:4] | 0x580[2] | afe_0xb6[2] | 0x582[2] | 0x583[2] | 0x584[2] | afe_0xb8[2] | 0x586[2] |
| PWM4/ I2C_CK/ pga_in<1>/PA[3] | GPIO | Rsvd | Rsvd | I2C_CK | PWM4 | 0x5a8[7:6] | 0x580[3] | afe_0xb6[3] | 0x582[3] | 0x583[3] | 0x584[3] | afe_0xb8[3] | 0x586[3] |
| PWM2/ I2C_SD/ PA[4] | GPIO | Rsvd | Rsvd | I2C_SD | PWM2 | 0x5a9[1:0] | 0x580[4] | afe_0xb6[4] | 0x582[4] | 0x583[4] | 0x584[4] | afe_0xb8[4] | 0x586[4] |
| PWM2_N/ I2C_CK/ I2C_MCK/ PA[5] | GPIO | Rsvd | I2C_MCK | I2C_CK | PWM2_N | 0x5a9[3:2] | 0x580[5] | afe_0xb6[5] | 0x582[5] | 0x583[5] | 0x584[5] | afe_0xb8[5] | 0x586[5] |
| PWM4_N/ I2C_SD/ I2C_MSD/ PA[6] | GPIO | Rsvd | I2C_MSD | I2C_SD | PWM4_N | 0x5a9[5:4] | 0x580[6] | afe_0xb6[6] | 0x582[6] | 0x583[6] | 0x584[6] | afe_0xb8[6] | 0x586[6] |
| PWM0/ TX_CYC2PA/ sar_ain<1>/PA[7] | GPIO | Rsvd | TX_CYC2PA | Rsvd | PWM0 | 0x5a9[7:6] | 0x580[7] | afe_0xb6[7] | 0x582[7] | 0x583[7] | 0x584[7] | afe_0xb8[7] | 0x586[7] |

Shared under NDA

| Pin | Default function | Pad Function Mux | | | | | GPIO Setting | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Register=3 | Register=2 | Register=1 | Register=0 | Register | Input (R) | IE | OEN | Output | Polarity | DS | Act as GPIO |
| PWM4/ MCN/ RX_CYC2LNA/ sar_ain<2>/PB[0] | GPIO | Rsvd | RX_CYC2LNA | MCN | PWM4 | 0x5aa[1:0] | 0x588[0] | afe_0xb9 [0] | 0x58a[0] | 0x58b[0] | 0x58c[0] | afe_0xbb [0] | 0x58e[0] |
| PWM1/ MDO/ TX_CYC2PA/ sar_ain<3>/PB[1] | GPIO | Rsvd | TX_CYC2PA | MDO | PWM1 | 0x5aa[3:2] | 0x588[1] | afe_0xb9 [1] | 0x58a[1] | 0x58b[1] | 0x58c[1] | afe_0xbb [1] | 0x58e[1] |
| PWM0/ MDI/ sar_ain<4>/PB[2] | GPIO | Rsvd | / | MDI | PWM0 | 0x5aa[5:4] | 0x588[2] | afe_0xb9 [2] | 0x58a[2] | 0x58b[2] | 0x58c[2] | afe_0xbb [2] | 0x58e[2] |
| PWM3_N/ MCK/ PWM2/ sar_ain<5>/PB[3] | GPIO | Rsvd | PWM2 | MCK | PWM3_N | 0x5aa[7:6] | 0x588[3] | afe_0xb9 [3] | 0x58a[3] | 0x58b[3] | 0x58c[3] | afe_0xbb [3] | 0x58e[3] |
| PWM2_N/ I2C_MCK/ UART_TX/ sar_ain<6>/pga_in<3>/PB[4] | GPIO | Rsvd | UART_TX | I2C_MCK | PWM2_N | 0x5ab[1:0] | 0x588[4] | afe_0xb9 [4] | 0x58a[4] | 0x58b[4] | 0x58c[4] | afe_0xbb [4] | 0x58e[4] |
| PWM4/ I2C_MSD/ UART_RX/ sar_ain<7>/pga_in<2>/PB[5] | GPIO | Rsvd | UART_RX | I2C_MSD | PWM4 | 0x5ab[3:2] | 0x588[5] | afe_0xb9 [5] | 0x58a[5] | 0x58b[5] | 0x58c[5] | afe_0xbb [5] | 0x58e[5] |
| PWM0_N/ I2C_MCK/ UART_RTS/ sar_ain<8>/PB[6] | GPIO | Rsvd | UART_RTS | I2C_MCK | PWM0_N | 0x5ab[5:4] | 0x588[6] | afe_0xb9 [6] | 0x58a[6] | 0x58b[6] | 0x58c[6] | afe_0xbb [6] | 0x58e[6] |
| PWM1/ I2C_MSD/ UART_CTS/ sar_ain<9>/PB[7] | GPIO | Rsvd | UART_CTS | I2C_MSD | PWM1 | 0x5ab[7:6] | 0x588[7] | afe_0xb9 [7] | 0x58a[7] | 0x58b[7] | 0x58c[7] | afe_0xbb [7] | 0x58e[7] |
| PC[0] | GPIO | / | / | / | / | 0x5ac[1:0] | 0x590[0] | 0x591[0] | 0x592[0] | 0x593[0] | 0x594[0] | 0x595[0] | 0x596[0] |
| PWM2_N/ PC[1] | GPIO | / | Rsvd | Rsvd | PWM2_N | 0x5ac[3:2] | 0x590[1] | 0x591[1] | 0x592[1] | 0x593[1] | 0x594[1] | 0x595[1] | 0x596[1] |

Shared under NDA

| Pin | Default function | Pad Function Mux | | | | | GPIO Setting | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Register=3 | Register=2 | Register=1 | Register=0 | Register | Input (R) | IE | OEN | Output | Polarity | DS | Act as GPIO |
| CN/ PWM1/ MCN/ UART_CTS/ PC[2] | CN | UART_CTS | MCN | PWM1 | CN | 0x5ac[5:4] | 0x590[2] | 0x591[2] | 0x592[2] | 0x593[2] | 0x594[2] | 0x595[2] | 0x596[2] |
| DO/ PWM0_N/ MDO/ UART_RTS/ PC[3] | DO | UART_RTS | MDO | PWM0_N | DO | 0x5ac[7:6] | 0x590[3] | 0x591[3] | 0x592[3] | 0x593[3] | 0x594[3] | 0x595[3] | 0x596[3] |
| DI/I2C_SD/ I2C_MSD/ MDI/ UART_TX/ PC[4] | DI | UART_TX | MDI | I2C_MSD | DI/I2C_SD | 0x5ad[1:0] | 0x590[4] | 0x591[4] | 0x592[4] | 0x593[4] | 0x594[4] | 0x595[4] | 0x596[4] |
| CK/I2C_CK/ I2C_MCK/ MCK/ UART_RX/ PC[5] | CK | UART_RX | MCK | I2C_MCK | CK/I2C_CK | 0x5ad[3:2] | 0x590[5] | 0x591[5] | 0x592[5] | 0x593[5] | 0x594[5] | 0x595[5] | 0x596[5] |
| PWM4/ PC[6] | GPIO | / | / | Rsvd | PWM4 | 0x5ad[5:4] | 0x590[6] | 0x591[6] | 0x592[6] | 0x593[6] | 0x594[6] | 0x595[6] | 0x596[6] |
| SWS/ PWM3/ sar_ain<0>/PC[7] | SWS | / | / | PWM3 | SWS | 0x5ad[7:6] | 0x590[7] | 0x591[7] | 0x592[7] | 0x593[7] | 0x594[7] | 0x595[7] | 0x596[7] |
| PWM1/ UART_CTS/ PWM0_N/ PD[0] | GPIO | / | PWM0_N | UART_CTS | PWM1 | 0x5ae[1:0] | 0x598[0] | 0x599[0] | 0x59a[0] | 0x59b[0] | 0x59c[0] | 0x59d[0] | 0x59e[0] |
| PWM0/ UART_RTS/ PWM1_N/ PD[1] | GPIO | / | PWM1_N | UART_RTS | PWM0 | 0x5ae[3:2] | 0x598[1] | 0x599[1] | 0x59a[1] | 0x59b[1] | 0x59c[1] | 0x59d[1] | 0x59e[1] |
| PWM3/ UART_TX/ PWM2/ PD[2] | GPIO | / | PWM2 | UART_TX | PWM3 | 0x5ae[5:4] | 0x598[2] | 0x599[2] | 0x59a[2] | 0x59b[2] | 0x59c[2] | 0x59d[2] | 0x59e[2] |

Shared under NDA

| Pin | Default function | Pad Function Mux | | | | | GPIO Setting | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Register=3 | Register=2 | Register=1 | Register=0 | Register | Input (R) | IE | OEN | Output | Polarity | DS | Act as GPIO |
| PWM0/ UART_RX/ TX_CYC2PA/ PD[3] | GPIO | / | TX_CYC2PA | UART_RX | PWM0 | 0x5ae[7:6] | 0x598[3] | 0x599[3] | 0x59a[3] | 0x59b[3] | 0x59c[3] | 0x59d[3] | 0x59e[3] |
| MSCN/ PE[0] | MSCN | / | / | / | MSCN | | 0x5a0[0] | 0x5a1[0] | 0x5a2[0] | 0x5a3[0] | 0x5a4[0] | 0x5a5[0] | 0x5a6[0] |
| MSDI/ PE[1] | MSDI | / | / | / | MSDI | | 0x5a0[1] | 0x5a1[1] | 0x5a2[1] | 0x5a3[1] | 0x5a4[1] | 0x5a5[1] | 0x5a6[1] |
| MSDO/ PE[2] | MSDO | / | / | / | MSDO | | 0x5a0[2] | 0x5a1[2] | 0x5a2[2] | 0x5a3[2] | 0x5a4[2] | 0x5a5[2] | 0x5a6[2] |
| MCLK/ PE[3] | MCLK | / | / | / | MCLK | | 0x5a0[3] | 0x5a1[3] | 0x5a2[3] | 0x5a3[3] | 0x5a4[3] | 0x5a5[3] | 0x5a6[3] |

***Notes:**

(1) All the registers in this table (IE, OEN, Register, Output, Input, DS, Act as GPIO, Polarity) can be set independently.

(2) IE: Input enable, high active. 1: enable input, 0: disable input.

(3) OEN: Output enable, low active. 0: enable output, 1: disable output.

(4) Register: Configure multiplexed functions in “Pad Function Mux” column.

(5) Output: configure GPO output.

(6) Input: read GPI input.

(7) DS: Drive strength. Default: 1.

(8) Act as GPIO: enable (1) or disable (0) GPIO function.

(9) Polarity: see **section 7.1.2 Connection relationship between GPIO and related modules**.

(10) Default function: By default, PC[2]~PC[5] are used as SPI Slave function, PC[7] is used as SWS function are used as MSPI function, while the other GPIOs are used as GPIO function.

(11) Priority: “Act as GPIO” has the highest priority. To configure as multiplexed function, disable GPIO function first.

(12) For all unused GPIOs, corresponding “IE” must be set as 0.

(13) When SWS/PC[7] “IE” is set as 1, this pin must be fixed as pull-up/pull-down state (float state is not allowed).

(14) afe_0xb6, afe_0xb8, afe_0xb9 and afe_0xbb marked in red color are analog registers; others are digital registers.

(15) Rsvd: reserved for internal use.

Shared under NDA

#### 7.1.1.2 Multiplexed functions

Each pin listed in Table 7-1 acts as the function in the "**Default Function**" column by default. By default, PC[2]~PC[5] are used as SPI Slave function, PC[7] is used as SWS function, while the other GPIOs are used as GPIO function.

If a pin with multiplexed functions does not act as GPIO function by default, to use it as GPIO, first set the bit in "**Act as GPIO**" column as 1b'1. After GPIO function is enabled, if the pin is used as output, both the bits in "**IE**" and "**OEN**" columns should be set as 1b'0, then set the register value in the "**Output**" column; if the pin is used as input, both the bits in "**IE**" and "**OEN**" columns should be set as 1b'1, and the input data can be read from the register in the "**Input**" column.

To use a pin as certain multiplexed function (neither the default function nor GPIO function), first clear the bit in "**Act as GPIO**" column to disable GPIO function, and then configure "**Register**" in "Pad Function Mux" column to enable multiplexed function correspondingly.

**Example 1:** PWM3/PWM0/RX_CYC2LNA/PA[0].

(1) This pin acts as GPIO function by default.

- ✧ To use this pin as general output, both address afe_0xb6[0] (IE) and 0x582[0] (OEN) should be set as 1b'0, then configure address 0x583[0] (Output).
- ✧ To use this pin as general input, both address afe_0xb6[0] (IE) and 0x582[0] (OEN) should be set as 1b'1, and the input data can be read from address 0x580[0] (Input).

(2) To use this pin as PWM3 function, address 0x586[0] (Act as GPIO) should be set as 1b'0, and 0x5a8[1:0] (Register) should be set as 2b'00.

(3) To use this pin as PWM0 function, address 0x586[0] (Act as GPIO) should be set as 1b'0, and 0x5a8[1:0] (Register) should be set as 2b'01.

(4) To use this pin as RX_CYC2LNA function, address 0x586[0] (Act as GPIO) should be set as 1b'0, and 0x5a8[1:0] (Register) should be set as 2b'10.

**Example 2:** CN/PWM1/MCN/UART_CTS/PC[2].

(1) This pin acts as SPI Slave CN function by default (0x596[2]=1b'0, 0x5ac[5:4]=2b'00).

(2) To use this pin as GPIO function, first set address 0x596[2] (Act as GPIO) as 1b'1.

- ✧ If the pin is used as general output, both address 0x591[2] (IE) and 0x592[2] (OEN) should be set as 1b'0, then configure address 0x593[2] (Output).
- ✧ If the pin is used as general input, both address 0x591[2] (IE) and 0x592[2] (OEN) should be set to 1b'1, and the input data can be read from address 0x590[2] (Input).

(3) To use it as PWM1 function, set address 0x596[2] (Act as GPIO) as 1b'0, and set 0x5ac[5:4] (Register) to 2b'01.

(4) To use it as SPI Master MCN function, set address 0x596[2] (Act as GPIO) as 1b'0, and set 0x5ac[5:4] (Register) to 2b'10.

(5) To use it as UART_CTS function, set address 0x596[2] (Act as GPIO) as 1b'0, and set 0x5ac[5:4] (Register) to 2b'11.

Shared under NDA

#### 7.1.1.3 Drive strength

The registers in the "DS" column are used to configure the corresponding pin's driving strength: "1" indicates maximum drive level, while "0" indicates minimal drive level.

The "DS" configuration will take effect when the pin is used as output. It's set as the strongest driving level by default. In actual applications, driving strength can be decreased to lower level if necessary.

PA[0] ~ PD[3] support drive strength up to 8mA (8mA when "DS"=1, 4mA when "DS"=0).

### 7.1.2 Connection relationship between GPIO and related modules

GPIO can be used to generate GPIO interrupt signal for interrupt system, counting or control signal for Timer/Counter module, or GPIO2RISC interrupt signal for interrupt system.

For the "Exclusive Or (XOR)" operation result for input signal from any GPIO pin and respective "Polarity" value, on one hand, it takes "And" operation with "irq" and generates GPIO interrupt request signal; on the other hand, it takes "And" operation with "m0/m1/m2", and generates counting signal in Mode 1 or control signal in Mode 2 for Timer0/Timer1/Timer2, or generates GPIO2RISC[0]/GPIO2RISC[1]/GPIO2RISC[2] interrupt request signal.

GPIO interrupt request signal = | ((input ^ polarity) & irq);

Counting (Mode 1) or control (Mode 2) signal for Timer0 = | ((input ^ polarity) & m0);

Counting (Mode 1) or control (Mode 2) signal for Timer1 = | ((input ^ polarity) & m1);

Counting (Mode 1) or control (Mode 2) signal for Timer2 = | ((input ^ polarity) & m2);

GPIO2RISC[0] interrupt request signal = | ((input ^ polarity) & m0);

GPIO2RISC[1] interrupt request signal = | ((input ^ polarity) & m1);

GPIO2RISC[2] interrupt request signal = | ((input ^ polarity) & m2).

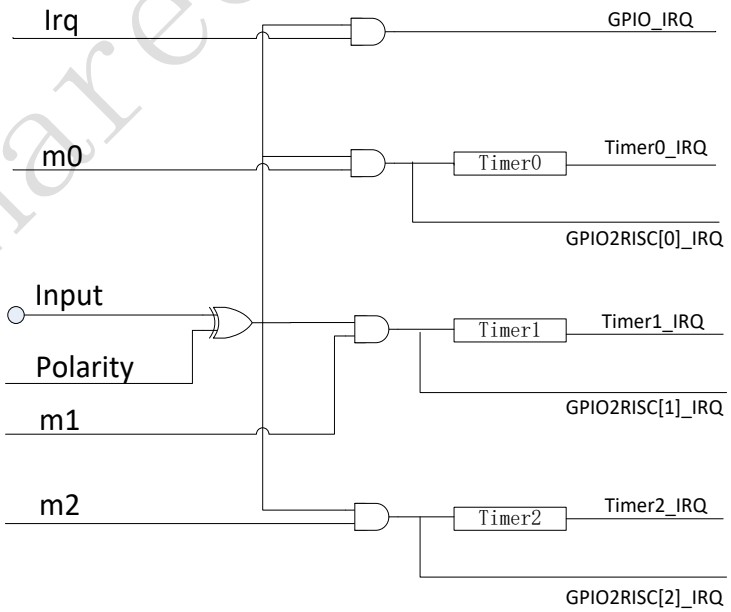



Figure 7- 1 Logic relationship between GPIO and related modules

Shared under NDA

Please refer to Table 7- 2 and Table 6- 1 to learn how to configure GPIO for interrupt system or Timer/Counter (Mode 1 or Mode 2).

(1) First enable GPIO function, enable IE and disable OEN. Please see section 7.1.1 Basic configuration.

(2) GPIO IRQ signal:

Select GPIO interrupt trigger edge (positive edge or negative edge) via configuring "**Polarity**", and set corresponding GPIO interrupt enabling bit "**Irq**".

Then set address 0x5b5[3] to enable GPIO IRQ.

Finally enable GPIO interrupt (irq_gpio) via MASK_2 (address 0x642[2]).

User can read addresses 0x5e0 ~ 0x5e3 to see which GPIO asserts GPIO interrupt request signal.

Note: 0x5e0[7:0]-->PA[7]~PA[0], 0x5e1[7:0]-->PB[7]~PB[0], 0x5e2[7:0]-->PC[7]~PC[0], PD[3]~PD[0]}.

Please ignore the GPIO herein which is not mentioned in the datasheet.

(3) Timer/Counter counting or control signal:

Configure "**Polarity**". In Timer Mode 1, it determines GPIO edge when Timer Tick counting increases. In Timer Mode 2, it determines GPIO edge when Timer Tick starts counting.

Then set "**m0/m1/m2**" to specify the GPIO which generates counting signal (Mode 1)/control signal (Mode 2) for Timer0/Timer1/Timer2.

User can read addresses 0x5e8~0x5eb/0x5f0~0x5f3/0x5f8~0x5fb to see which GPIO asserts counting signal (in Mode 1) or control signal (in Mode 2) for Timer0/Timer1/Timer2.

Note: Timer0: 0x5e8[7:0]-->PA[7]~PA[0], 0x5e9[7:0]-->PB[7]~PB[0], 0x5ea[7:0]-->PC[7]~PC[0], PD[3]~PD[0]}; Timer1: 0x5f0[7:0]-->PA[7]~PA[0], 0x5f1[7:0]-->PB[7]~PB[0], 0x5f2[7:0]-->PC[7]~PC[0], PD[3]~PD[0]}; Timer2: 0x5f8[7:0]-->PA[7]~PA[0], 0x5f9[7:0]-->PB[7]~PB[0], 0x5fa[7:0]-->PC[7]~PC[0], PD[3]~PD[0]}.

(4) GPIO2RISC IRQ signal:

Select GPIO2RISC interrupt trigger edge (positive edge or negative edge) via configuring "**Polarity**", and set corresponding GPIO enabling bit **"m0"/"m1"/"m2"**.

Enable GPIO2RISC[0]/GPIO2RISC[1]/GPIO2RISC[2] interrupt via MASK_2, i.e. "gpio2risc[0]" (address 0x642[5]) / "gpio2risc[1]" (address 0x642[6]) / "gpio2risc[2]" (address 0x642[7]).

Shared under NDA

Table 7- 2GPIO lookup table2

| Pin | Input (R) | Polarity 1: active low 0: active high | Irq | m0 | m1 | m2 |
|---|---|---|---|---|---|---|
| PA[0] | 0x580[0] | 0x584[0] | 0x587[0] | 0x5b8[0] | 0x5c0[0] | 0x5c8[0] |
| PA[1] | 0x580[1] | 0x584[1] | 0x587[1] | 0x5b8[1] | 0x5c0[1] | 0x5c8[1] |
| PA[2] | 0x580[2] | 0x584[2] | 0x587[2] | 0x5b8[2] | 0x5c0[2] | 0x5c8[2] |
| PA[3] | 0x580[3] | 0x584[3] | 0x587[3] | 0x5b8[3] | 0x5c0[3] | 0x5c8[3] |
| PA[4] | 0x580[4] | 0x584[4] | 0x587[4] | 0x5b8[4] | 0x5c0[4] | 0x5c8[4] |
| PA[5] | 0x580[5] | 0x584[5] | 0x587[5] | 0x5b8[5] | 0x5c0[5] | 0x5c8[5] |
| PA[6] | 0x580[6] | 0x584[6] | 0x587[6] | 0x5b8[6] | 0x5c0[6] | 0x5c8[6] |
| PA[7] | 0x580[7] | 0x584[7] | 0x587[7] | 0x5b8[7] | 0x5c0[7] | 0x5c8[7] |
| PB[0] | 0x588[0] | 0x58c[0] | 0x58f[0] | 0x5b9[0] | 0x5c1[0] | 0x5c9[0] |
| PB[1] | 0x588[1] | 0x58c[1] | 0x58f[1] | 0x5b9[1] | 0x5c1[1] | 0x5c9[1] |
| PB[2] | 0x588[2] | 0x58c[2] | 0x58f[2] | 0x5b9[2] | 0x5c1[2] | 0x5c9[2] |
| PB[3] | 0x588[3] | 0x58c[3] | 0x58f[3] | 0x5b9[3] | 0x5c1[3] | 0x5c9[3] |
| PB[4] | 0x588[4] | 0x58c[4] | 0x58f[4] | 0x5b9[4] | 0x5c1[4] | 0x5c9[4] |
| PB[5] | 0x588[5] | 0x58c[5] | 0x58f[5] | 0x5b9[5] | 0x5c1[5] | 0x5c9[5] |
| PB[6] | 0x588[6] | 0x58c[6] | 0x58f[6] | 0x5b9[6] | 0x5c1[6] | 0x5c9[6] |
| PB[7] | 0x588[7] | 0x58c[7] | 0x58f[7] | 0x5b9[7] | 0x5c1[7] | 0x5c9[7] |
| PB[0] | 0x588[0] | 0x58c[0] | 0x58f[0] | 0x5b9[0] | 0x5c1[0] | 0x5c9[0] |
| PC[0] | 0x590[0] | 0x594[0] | 0x597[0] | 0x5ba[0] | 0x5c2[0] | 0x5ca[0] |
| PC[1] | 0x590[1] | 0x594[1] | 0x597[1] | 0x5ba[1] | 0x5c2[1] | 0x5ca[1] |
| PC[2] | 0x590[2] | 0x594[2] | 0x597[2] | 0x5ba[2] | 0x5c2[2] | 0x5ca[2] |
| PC[3] | 0x590[3] | 0x594[3] | 0x597[3] | 0x5ba[3] | 0x5c2[3] | 0x5ca[3] |
| PC[4] | 0x590[4] | 0x594[4] | 0x597[4] | 0x5ba[4] | 0x5c2[4] | 0x5ca[4] |
| PC[5] | 0x590[5] | 0x594[5] | 0x597[5] | 0x5ba[5] | 0x5c2[5] | 0x5ca[5] |
| PC[6] | 0x590[6] | 0x594[6] | 0x597[6] | 0x5ba[6] | 0x5c2[6] | 0x5ca[6] |
| PC[7] | 0x590[7] | 0x594[7] | 0x597[7] | 0x5ba[7] | 0x5c2[7] | 0x5ca[7] |
| PD[0] | 0x598[0] | 0x59c[0] | 0x59f[0] | 0x5bb[0] | 0x5c3[0] | 0x5cb[0] |
| PD[1] | 0x598[1] | 0x59c[1] | 0x59f[1] | 0x5bb[1] | 0x5c3[1] | 0x5cb[1] |
| PD[2] | 0x598[2] | 0x59c[2] | 0x59f[2] | 0x5bb[2] | 0x5c3[2] | 0x5cb[2] |
| PD[3] | 0x598[3] | 0x59c[3] | 0x59f[3] | 0x5bb[3] | 0x5c3[3] | 0x5cb[3] |
| PE[0] | 0x5a0[0] | 0x5a4[0] | 0x5a7[0] | 0x5bb[4] | 0x5c3[4] | 0x5cb[4] |
| PE[1] | 0x5a0[1] | 0x5a4[1] | 0x5a7[1] | 0x5bb[5] | 0x5c3[5] | 0x5cb[5] |
| PE[2] | 0x5a0[2] | 0x5a4[2] | 0x5a7[2] | 0x5bb[6] | 0x5c3[6] | 0x5cb[6] |
| PE[3] | 0x5a0[3] | 0x5a4[3] | 0x5a7[3] | 0x5bb[7] | 0x5c3[7] | 0x5cb[7] |

Shared under NDA

### 7.1.3 Pull-up/Pull-down resistor

The GPIOs including PA[5]~PA[7] and PB[0]~PB[7] support configurable pull-up resistor of rank x1 and x100 or pull-down resistor of rank x10 which are all disabled by default. Analog registers afe_0x08~afe_0x0a[5:0] serve to control the pull-up/pull-down resistor for each of these GPIOs.

The GPIOs including PA[0]~PA[4], PC[0]~PC[7] and PD[0]~PD[3] support pull-down resistor of rank x10 which are all disabled by default. Analog registers including afe_0x0a[7:6], afe_0x0b[7:0] and afe_0x0c[6:0] serve to control the pull-down resistor for each of these GPIOs.

Please refer to Table 7-3 for details.

Take the PB[0] for an example: Setting analog register afe_0x08[1:0] to 2b'01/2b'10/2b'11 is to respectively enable pull-up resistor of rank x100 / pull-up resistor of rank x1 / pull-down resistor of rank x10 for PB[0]; Clearing the two bits (default value) disables pull-up and pull-down resistor for PB[0].

Take the PA[0] for another example: Setting analog register afe_0x0a[6] to 1b'1 is to enable pull-down resistor of rank x10 for PA[0]; Clearing the bit (default value) disables pull-down resistor for PA[0].

Table 7- 3 Analog registers for pull-up/pull-down resistor control

| Address | Mnemonic | Default | Description |
|---|---|---|---|
| afe_0x08<1:0> | pullupdown_ctrl<1:0> | 00 | PB[0] pull up/down control<br>00 – No pull up/down resistor<br>01 –x100 pull-up resistor<br>10 – x1 pull-up resistor<br>11 – x10 pull-down resistor |
| afe_0x08<3:2> | pullupdown_ctrl<1:0> | 00 | PB[1] pull up/down control<br>00 – No pull up/down resistor<br>01 –x100 pull-up resistor<br>10 – x1 pull-up resistor<br>11 – x10 pull-down resistor |
| afe_0x08<5:4> | pullupdown_ctrl<1:0> | 00 | PB[2] pull up/down control<br>00 – No pull up/down resistor<br>01 – x100 pull-up resistor<br>10 – x1 pull-up resistor<br>11 – x10 pull-down resistor |
| afe_0x08<7:6> | pullupdown_ctrl<1:0> | 00 | PB[3] pull up/down control<br>00 – No pull up/down resistor<br>01 – x100 pull-up resistor<br>10 – x1 pull-up resistor<br>11 – x10 pull-down resistor |
| afe_0x09<1:0> | pullupdown_ctrl<1:0> | 00 | PB[4] pull up/down control<br>00 – No pull up/down resistor<br>01 – x100 pull-up resistor<br>10 – x1 pull-up resistor<br>11 – x10 pull-down resistor |

Shared under NDA

| Address | Mnemonic | Default | Description |
|---|---|---|---|
| afe_0x09<3:2> | pullupdown_ctrl<1:0> | 00 | PB[5] pull up/down control<br>00 – No pull up/down resistor<br>01 – x100 pull-up resistor<br>10 – x1 pull-up resistor<br>11 – x10 pull-down resistor |
| afe_0x09<5:4> | pullupdown_ctrl<1:0> | 00 | PB[6] pull up/down control<br>00 – No pull up/down resistor<br>01 – x100 pull-up resistor<br>10 – x1 pull-up resistor<br>11 – x10 pull-down resistor |
| afe_0x09<7:6> | pullupdown_ctrl<1:0> | 00 | PB[7] pull up/down control<br>00 – No pull up/down resistor<br>01 – x100 pull-up resistor<br>10 – x1 pull-up resistor<br>11 – x10 pull-down resistor |
| afe_0x0a<1:0> | pullupdown_ctrl<1:0> | 00 | PA[5] pull up/down control<br>00 – No pull up/down resistor<br>01 – x100 pull-up resistor<br>10 – x1 pull-up resistor<br>11 – x10 pull-down resistor |
| afe_0x0a<3:2> | pullupdown_ctrl<1:0> | 00 | PA[6] pull up/down control<br>00 – No pull up/down resistor<br>01 – x100 pull-up resistor<br>10 – x1 pull-up resistor<br>11 – x10 pull-down resistor |
| afe_0x0a<5:4> | pullupdown_ctrl<1:0> | 00 | PA[7] pull up/down control<br>00 – No pull up/down resistor<br>01 – x100 pull-up resistor<br>10 – x1 pull-up resistor<br>11 – x10 pull-down resistor |
| afe_0x0a<7:6> | pulldown_ctrl<1:0> | 00 | PA[1]~PA[0] pull down control<br>0– No pull down resistor<br>1 – Enable x10 pull-down resistor |
| afe_0x0b<7:0> | pulldown_ctrl<7:0> | 0x00 | {PC[4]~ PC[0], PA[4]~ PA[2]} pull down control<br>0– No pull down resistor<br>1 – Enable x10 pull-down resistor |
| afe_0x0c<6:0> | pulldown_ctrl<6:0> | 000000 | {PD[3]~ PD[0], PC[7]~ PC[5]} pull down control<br>0– No pull down resistor<br>1 – Enable x10 pull-down resistor |

Shared under NDA

## 7.2 SWS

The TLSR8367 supports Single Wire Slave (SWS) interface for debugging. SWS represents the Slave device of the single wire communication system developed by Telink. The maximum data rate can be up to 2Mbps.

SWS usage is not supported in power-saving mode (deep sleep or suspend). The TLSR8367 has to be waked up by using IO or 32k RC wakeup, so that it can respond to the Swire commands for debug/programming.

## 7.3 I2C

The TLSR8367 embeds I2C hardware module, which could act as Master mode or Slave mode. I2C is a popular inter-IC interface requiring only 2 bus lines, a serial data line (SDA) and a serial clock line (SCL).

I2CSCT (address 0x03) bit[1] and bit[4] serves to select I2C Master mode or Slave mode. By default, 0x03 bit[4] is set as 1b'1 and bit[1] is set as 1b'0, therefore I2C module of the TLSR8367 acts as Slave mode by default.

### 7.3.1 Communication protocol

Telink I2C module supports standard mode (100kbps) and Fast-mode (400kbps) with restriction that system clock must be by at least 10x of data rate.

Two wires, SDA and SCL (SCK) carry information between Master device and Slave device connected to the bus. Each device is recognized by unique address (ID). Master device is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. Slave device is the device addressed by a Master.

Both SDA and SCL are bidirectional lines connected to a positive supply voltage via a pull-up resister. It's recommended to use external 3.3kohm pull-up resistor. For standard mode, the internal pull-up resistor of rank x1 can be used instead of the external 3.3kohm pull-up.

When the bus is free, both lines are HIGH. It's noted that data in SDA line must keep stable when clock signal in SCL line is at high level, and level state in SDA line is only allowed to change when clock signal in SCL line is at low level.

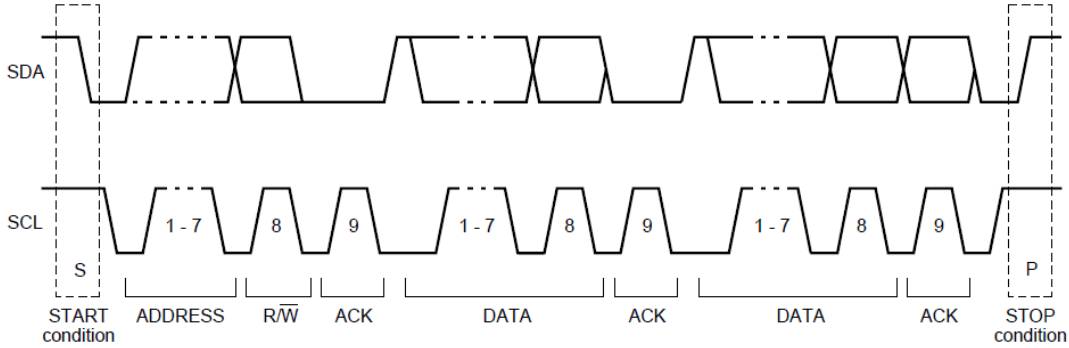



Figure 7- 2 I2C timing chart

Shared under NDA

### 7.3.2 Register table

Table 7- 4 Register configuration for I2C

| Address | Name | R/W | Description | Reset Value |
|---|---|---|---|---|
| 0x00 | I2CSP | RW | I2C master clock speed | 0x1f |
| 0x01 | I2CMID | RW | [7:1] I2C master ID | 0x5c |
| 0x02 | I2CMST | RW | [0]: master busy<br>[1]: master packet busy<br>[2]: master received status<br>0 for ACK; 1 for NAK | 0x00 |
| 0x03 | I2CSCT | RW | [0]: address auto increase enable<br>[1]: I2C master enable (1)<br>[2]: sub-mode select in I2C slave mode<br>0- DMA mode<br>1- Mapping Mode<br>[4]: I2C slave enable (1) | 0x11 |
| 0x04 | I2CAD | RW | [7:0] Data buffer in master mode | 0x5a |
| 0x05 | I2CDW | RW | [7:0] Data buffer in master mode | 0xf1 |
| 0x06 | I2CDR | RW | [7:0] Data buffer for Read or Write in master mode | 0x00 |
| 0x07 | I2CCLT | RW | [0]: launch ID cycle<br>[1]: launch address cycle<br>(send I2CAD data)<br>[2]: launch data write cycle<br>[3]: launch data read cycle<br>For Master Write: 0: I2CAD&I2CDW,<br>1: I2CAD&I2CDW&I2CDR)<br>To write 3 bytes: bit[3]=1;<br>To write 2 bytes: bit[3]=0.<br>For Master Read: always 1.<br>[4]: launch start cycle<br>[5]: launch stop cycle<br>[6]: enable read ID<br>[7]: enable ACK in read command | 0x00 |
| 0x20 | ADROFFSET | RO | [6:0] mapped host address offset | 0x00 |
| 0x21 | HOSTIRQ | RO | [0]: host cmd irq flag, I2C host operation have happened. Write 1 to clear.<br>[1]: host read flag, I2C host operation have happened and is read operation. Write 1 to clear<br>([2]: software irq flag, write 1 to clear<br>[3]: software irq, write 1 to set | 0x00 |
| 0x22 | MAPADRL | R/W | Low byte of Mapping mode buffer address | 0x80 |
| 0x23 | MAPADRH | R/W | High byte of Mapping mode buffer address | 0x9f |

Shared under NDA

### 7.3.3 I2C Slave mode

I2C module of the TLSR8367 acts as Slave by default (Address 0x03[4] should be set as 1b'1 to enable I2C Slave mode).

I2C slave address can be configured via register I2CID (address 0x01) [7:1].

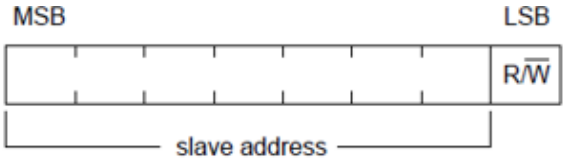



Figure 7- 3 Byte consisted of slave address and R/W flag bit

I2C slave mode supports two sub modes including Direct Memory Access (DMA) mode and Mapping mode, which is selectable via I2CSCT (address 0x03) bit[2].

In I2C Slave mode, Master could initiate transaction anytime. I2C slave module will reply with ACK automatically. To monitor the start of I2C transaction, user could set interrupt from GPIO for SCA or SCL.

#### 7.3.3.1 DMA mode

By default, I2CSCT (address 0x03) bit[2] is set as 1b'0, therefore DMA mode is selected by default.

In DMA mode, other devices (Master) could access (read/write) designated address in Register and/or SRAM of the TLSR8369 according to I2C protocol. I2C module of the TLSR8367 will execute the read/write command from I2C Master automatically. But user needs to notice that the system clock shall be at least 10x faster than I2C bit rate.

The access address designated by Master is offset by 0x800000. In the TLSR8367, Register address starts from 0x800000 and SRAM address starts from 0x808000. For example, if Addr High(AddrH) is 0xaa and Addr Low (AddrL) is 0xcc, the real address of accessed data is 0x80aacc.

In DMA mode, Master could read/write data byte by byte. The designated access address is initial address and it supports auto increment by setting I2CSCT (address 0x03) bit[0] to 1b'1.

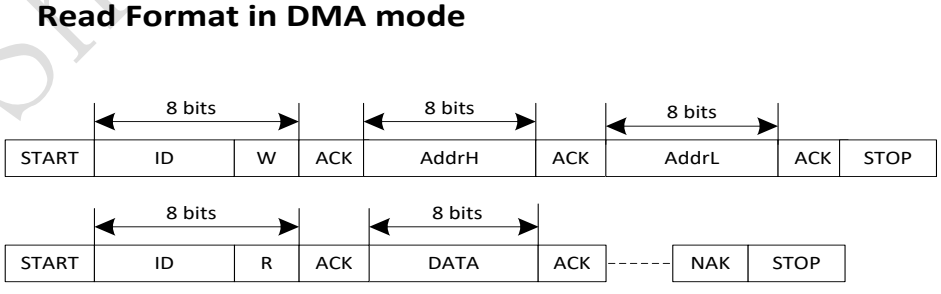



Figure 7- 4 Read format in DMA mode

Shared under NDA

**Write Format in DMA mode**

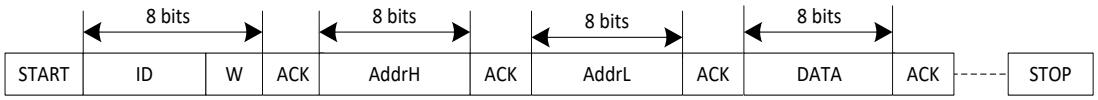



Figure 7- 5 Write format in DMA mode

#### 7.3.3.2 Mapping mode

Mapping mode could be enabled via setting register I2CSCT (address 0x03) bit[2] as 1b'1.

In Mapping mode, data written and read by I2C master will be redirected to specified 128-byte buffer in SRAM. User could specify the initial address of the buffer by configuring registers MAPADRL (address 0x22, lower byte) and MAPADRH (address 0x23, higher byte). The first 64-byte buffer is for written data and following 64-byte buffer is for read data. Every time the data access will start from the beginning of the Write-buffer/Read-buffer after I2C stop condition occurs. The last accessed data address could be checked in register ADROFFSET (address 0x20) [6:0] which is only updated after I2C STOP occurs.

**Read Format in mapping mode**

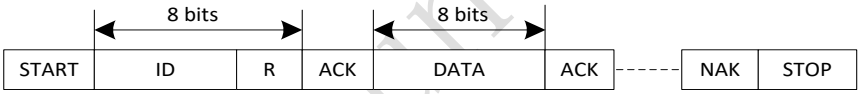



Figure 7- 6 Read format in Mapping mode

**Write Format in mapping mode**

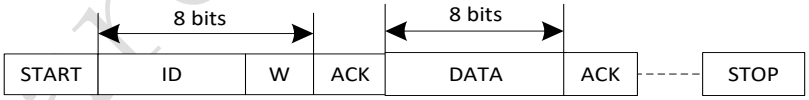



Figure 7- 7 Write format in Mapping mode

#### 7.3.4 I2C Master mode

I2CSCT (address 0x03) bit[1] should be set as 1b'1 to enable I2C master mode for the TLSR8367.

Address 0x00 serves to set I2C Master clock: $F_{I2C}$ = System Clock / (4 *clock speed configured in address 0x00).

A complete I2C protocol contains START, Slave Address, R/W bit, data, ACK and STOP. Slave address could be configured via address 0x01[7:1].

I2C Master (i.e. I2C module of the TLSR8367) could send START, Slave Address, R/W bit, data and STOP cycle by configuring address 0x07. I2C master will send enabled cycles in the correct sequence.

Shared under NDA

Address 0x02 serves to indicate whether Master/Master packet is busy, as well as Master received status. Bit[0] will be set to 1 when one byte is being sent, and this bit can be automatically cleared after a start signal/ address byte/acknowledge signal/data /stop signal is sent. Bit[1] is set to 1 when the start signal is sent, and this bit will be automatically cleared after the stop signal is sent. Bit[2] indicates whether to succeed in sending acknowledgement signal.

#### 7.3.4.1 I2C Master Write transfer

I2C Master has 3-byte buffer for write data, which are I2CAD (0x04), I2CDW (0x05) and I2CDR (0x06). Write transfer will be completed by I2C master module.

For example, to implement an I2C write transfer with 3-byte data, which contains START, Slave Address, Write bit, ack from Slave, 1st byte, ack from slave, 2nd byte, ack from slave, 3rd byte, ack from slave and STOP, user needs to configure I2C slave address to I2CID (0x01) [7:1], 1st byte data to I2CAD, 2nd byte data to I2CDW and 3rd byte to I2CDR. To start I2C write transfer, I2CCLT (0x07) is configured to 0x3f (0011 1111). I2C Master will launch START, Slave address, Write bit, load ACK to I2CMST (0x02) [2], send I2CAD data, load ACK to I2CMST[2], send I2CDW data, load ACK to I2CMST[2], send I2CDR data, load ACK to I2CMST[2] and then STOP sequentially.

For I2C write transfer whose data are more than 3 bytes, user could split the cycles according to I2C protocol.

#### 7.3.4.2 I2C Master Read transfer

I2C Master has one byte buffer for read data, which is I2CDR (0x06). Read transfer will be completed by I2C Master.

For example, to implement an I2C read transfer with 1 byte data, which contains START, Slave Address, Read bit, Ack from Slave, 1$^{st}$ byte from Slave, Ack by master and STOP, user needs to configure I2C slave address to I2CID (0x01) [7:1]. To start I2C read transfer, I2CCLT (0x07) is configured to 0xf9 (1111 1001). I2C Master will launch START, Slave address, Read bit, and load ACK to I2CMST (0x02) [2], load data to I2CDR, reply ACK and then STOP sequentially.

For I2C read transfer whose data are more than 1 byte, user could split the cycles according to I2C protocol.

### 7.3.5 I2C and SPI Usage

I2C hardware and SPI hardware modules in the chip share part of the hardware, as a result, when both hardware interfaces are used, the restrictions listed within this section need to be taken into consideration.

I2C and SPI hardware cannot be used as Slave at the same time.

The other cases are supported, including:

- ✧ I2C and SPI can be used as Master at the same time.
- ✧ I2C Master and SPI Slave can be used at the same time.
- ✧ I2C Slave and SPI Master can be used at the same time.

Shared under NDA

## 7.4 SPI

The TLSR8367 embeds SPI (Serial Peripheral interface), which could act as Master mode or Slave mode. SPI is a high-speed, full-duplex and synchronous communication bus requiring 4 bus lines including a chip select (CS) line, a data input (DI) line, a data output (DO) line and a clock (CK) line.

Register SPICT (address 0x09) bit[1] and bit[6] serve to select SPI Master mode or Slave mode. By default, 0x09 bit[1] is set as 1b'0 and bit[6] is set as 1b'1, therefore SPI acts as Slave mode by default.

### 7.4.1 Register table

Table 7- 5 Register configuration for SPI

| Address | Name | R/W | Description | Reset Value |
|---|---|---|---|---|
| 0x08 | SPIDAT | RW | SPI data access | 00 |
| 0x09 | SPICT | RW | [0]: p_csn<br>[1]: enable SPI master mode (1)<br>[2]: spi data output disable<br>[3]: 1 for read command; 0 for write command<br>[4]: address auto increase<br>[5]: share_mode<br>[6]: enable SPI slave mode (1)<br>[7]: busy status | 51 |
| 0x0a | SPISP | RW | [6:0]: SPI clock speed<br>[7]: SPI function mode, p_csn, p_scl, p_sda and p_sdo function as SPI if 1 | 05 |
| 0x0b | SPIMODE | RW | [0]: inverse SPI clock output<br>[1]: delay half clk for SPI data output | 00 |

Shared under NDA

### 7.4.2 SPI Master mode

Address 0x09 bit[1] should be set as 1b'1 to enable SPI Master mode.

Register SPISP (address 0x0a) serves to configure SPI pin and clock: setting 0x0a bit[7] as 1b'1 is to enable SPI function mode, and corresponding pins can be used as SPI pins; SPI clock = system clock/((clock speed configured in address 0x0a bit[6:0] +1)*2).

SPIDAT (address 0x08) serves as the data register. One reading/writing operation of 0x08 enables the SPI_CK pin to generate 8 SPI clock cycles.

Telink SPI supports four standard working modes: Mode 0~Mode 3. Register SPIMODE (address 0x0b) serves to select one of the four SPI modes:

Table 7- 6 SPI Master mode

| SPI mode | CPOL/CPHA | SPIMODE register (Address 0x0b) |
|---|---|---|
| Mode 0 | CPOL=0, CPHA=0 | bit[0]=0, bit[1]=0 |
| Mode 1 | CPOL=0, CPHA=1 | bit[0]=0, bit[1]=1 |
| Mode 2 | CPOL=1, CPHA=0 | bit[0]=1, bit[1]=0 |
| Mode 3 | CPOL=1, CPHA=1 | bit[0]=1, bit[1]=1 |
| CPOL: Clock Polarity<br>When CPOL=0, SPI_CLK keeps low level in idle state;<br>When CPOL=1, SPI_CLK keeps high level in idle state.<br>CPHA: Clock Phase<br>When CPHA=0, data is sampled at the first edge of clock period<br>When CPHA=1, data is sampled at the latter edge of clock period | | |

Address 0x09 bit[0] serves to control the CS line: when the bit is set to 1, the CS level is high; when the bit is cleared, the CS level is low.

Address 0x09 bit[2] is the disabling bit for SPI Master output. When the bit is cleared, MCU writes data into address 0x08, then the SPI_DO pin outputs the data bit by bit during the 8 clock cycles generated by the SPI_CK pin. When the bit is set to 1b'1, SPI_DO output is disabled.

Address 0x09 bit[3] is the enabling bit for SPI Master reading data function. When the bit is set to 1b'1, MCU reads the data from address 0x08, then the input data from the SPI_DI pin is shifted into address 0x08 during the 8 clock cycles generated by the SPI_CK pin. When the bit is cleared, SPI Master reading function is disabled.

Address 0x09[5] is the enabling bit for share mode, i.e. whether SPI_DI and SPI_DO share one common line.

Users can read address 0x09 bit[7] to get SPI busy status, i.e. whether the 8 clock pulses have been sent.

Shared under NDA

### 7.4.3 SPI Slave mode

SPI for the TLSR8367 acts as Slave mode by default. (Address 0x09 bit[6] should be set as 1b'1 to enable SPI Slave mode.)

SPI Slave mode supports DMA. User could access registers of the TLSR8367 by SPI interface. It's noted that system clock of TLSR8367 shall be at least 5x faster than SPI clock for reliable connection. Address 0x0a should be written with data 0xa5 by the SPI host to activate SPI Slave mode. SPI salve only supports Mode0 and Mode3.

Table 7- 7 SPI Slave mode

| SPI slave mode | CPOL/CPHA |
|---|---|
| Mode 0 | CPOL=0, CPHA=0 |
| Mode 3 | CPOL=1, CPHA=1 |
| Receive data at positive edge of SPI MCLK clock.<br>Send data at negative edge of SPI MCLK clock. | |

Address 0x09[4] is dedicated for SPI Slave mode and indicates address auto increment. SPI write command format and read command format are illustrated in Figure 7-8:

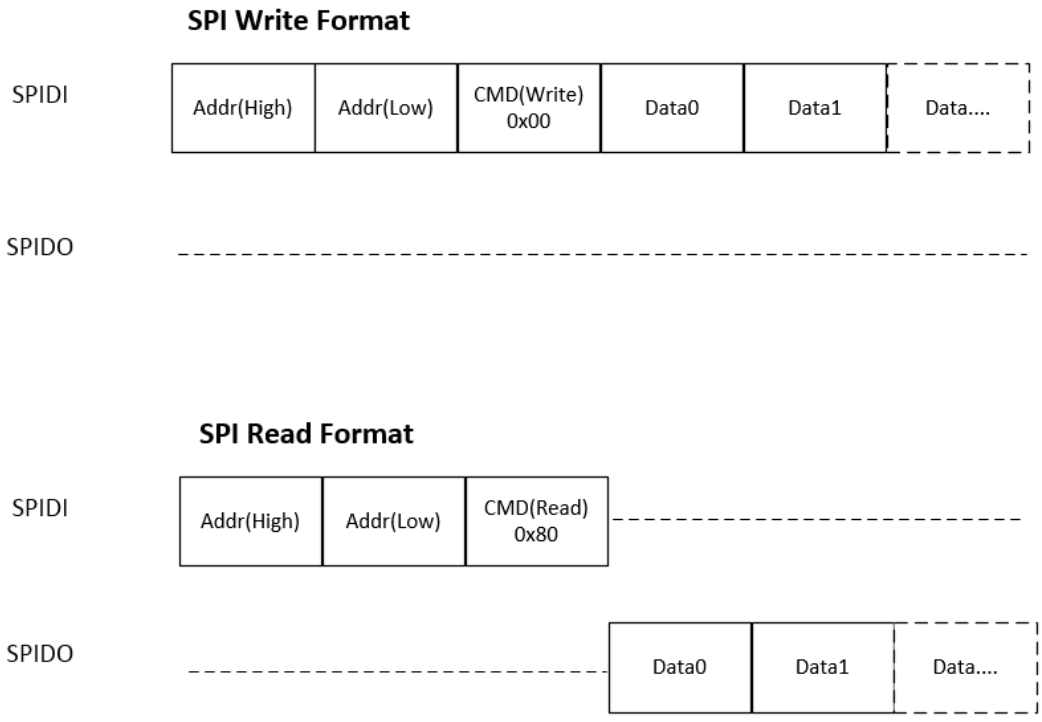



Figure 7- 8 SPI write/read command format

### 7.4.4 I2C and SPI Usage

I2C hardware and SPI hardware modules in the chip share part of the hardware, as a result, when both hardware interfaces are used, certain restrictions apply. See **Section 7.3.5 I2C and SPI Usage** for detailed instructions.

## 7.5 UART

The TLSR8367 embeds UART (Universal Asynchronous Receiver/Transmitter) to implement full-duplex transmission and reception via UART TX and RX interface. Both TX and RX interface are 4-layer FIFO (First In First Out) interface.

Hardware flow control is supported via RTS and CTS.

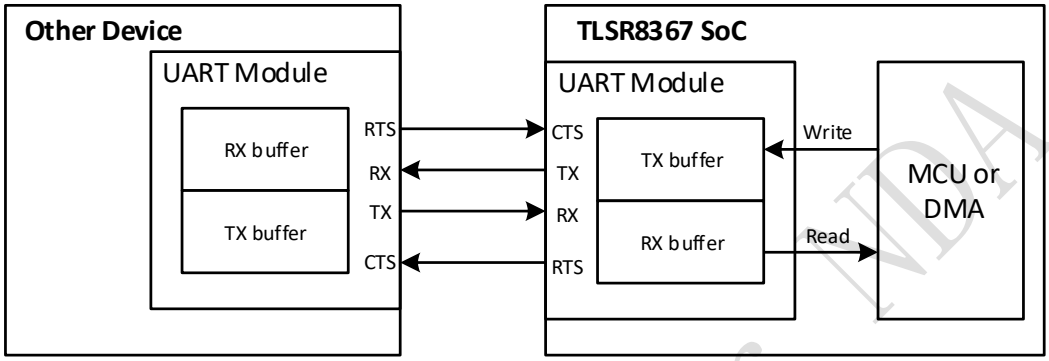



Figure 7- 9 UART communication

As shown in Figure 7-9, data to be sent is first written into TX buffer by MCU or DMA, then UART module transmits the data from TX buffer to other device via pin TX. Data to be read from other device is first received via pin RX and sent to RX buffer, then the data is read by MCU or DMA.

If RX buffer of the TLSR8367 UART is close to full, the TLSR8367 will send a signal (configurable high or low level) via pin RTS to inform other device that it should stop sending data. Similarly, if the TLSR8367 receives a signal from pin CTS, it indicates that RX buffer of other device is close to full and the TLSR8367 should stop sending data.

Table 7- 8 Register configuration for UART

| Address | Name | R/W | Description | Reset Value |
|---|---|---|---|---|
| 0x90 | uart_data_buf0 | R/W | write/read buffer[7:0] | *No power on reset |
| 0x91 | Uart_data_buf1 | R/W | Write/read buffer[15:8] | |
| 0x92 | Uart_data_buf2 | RW | Write/read buffer[23:16] | |
| 0x93 | Uart_data_buf3 | R/W | Write/read buffer[31:24] | |
| 0x94 | uart_clk_div[7:0] | RW | uart clk div register: | 0xff |
| 0x95 | Uart_clk_div[15:8] | R/W | uart_sclk = sclk/(uart_clk_div[14:0]+1)<br>uart_clk_div[15] :<br>1: enable clock divider, 0: disable. | 0x0f |
| 0x96 | Uart ctrl0 | R/W | [3:0] bwpc, bit width, should be larger than 2<br>Baudrate = uart_sclk/(bwpc+1)<br>[4] rx dma enable<br>[5] tx dma enable<br>[6] rx interrupt enable<br>[7]tx interrupt enable | 0x0f |
| 0x97 | Uart_ctrl1 | R/W | [0] cts select, 0: cts_i, 1: cts _i inverter | 0x0e |

Shared under NDA

| Address | Name | R/W | Description | Reset Value |
|---|---|---|---|---|
| | | | [1]:cts enable, 1: enable, 0, disable<br>[2]:Parity, 1: enable, 0 :disable<br>[3]: even Parity or odd<br>[5:4]: stop bit<br>00: 1 bit, 01: 1.5bit, 1x: 2bits<br>[6]: ttl<br>[7]: uart tx, rx loopback | |
| 0x98 | Uart_ctrl2 | R/W | [3:0] rts trig level<br>[4] rts Parity<br>[5] rts manual value<br>[6] rts manual enable<br>[7] rts enable | 0xa5 |
| 0x99 | Uart_ctrl3 | R/W | [3:0]: rx_irq_trig level<br>[7:4] tx_irq_trig level | 0x44 |
| 0x9a | R_rxtimeout_o[7:0] | R/W | The setting is transfer one bytes need cycles base on uart_clk. For example, if transfer one bytes (1 start bit+8bits data+1 priority bit+2 stop bits) total 12 bits, this register setting should be (bwpc+1)*12. | 0x0f |
| 0x9b | R_rxtimeout_o[9:8] | R/W | 2'b00:rx timeout time is r_rxtimeout[7:0]<br>2'b01:rx timeout time is r_rxtimeout[7:0]*2<br>2'b10:rx timeout time is r_rxtimeout[7:0]*3<br>3'b11: rx timeout time is r_rxtimeout[7:0]*4<br>R_rxtimeout is for rx dma to decide the end of each transaction. Supposed the interval between each byte in one transaction is very short. | 0x00 |
| 0x9c | Buf_cnt | R | [3:0]: r_buf_cnt<br>[7:4]: t_buf_cnt | 0x00 |
| 0x9d | Uart_sts | R | [2:0] rbcnt<br>[3] irq<br>[6:4]wbcnt<br>[6] write 1 clear rx<br>[7] rx_err, write 1 clear tx | 0x00 |

***Note:** Addresses 0x90~0x93 won't be reset after power on.

Shared under NDA

Addresses 0x90~0x93 serve to write data into TX buffer or read data from RX buffer.

Addresses 0x94~0x95 serve to configure UART clock.

Address 0x96 serves to set baud rate (bit[3:0]), enable RX/TX DMA mode (bit[4:5]), and enable RX/TX interrupt (bit[6:7]).

Address 0x97 mainly serves to configure CTS. Bit[1] should be set to 1b'1 to enable CTS. Bit[0] serves to configure CTS signal level. Bit[2:3] serve to enable parity bit and select even/odd parity. Bit[5:4] serve to select 1/1.5/2 bits for stop bit. Bit[6] serves to configure whether RX/TX level should be inverted.

Address 0x98 serves to configure RTS. Bit[7] and Bit[3:0] serve to enable RTS and configure RTS signal level.

Address 0x99 serves to configure the number of bytes in RX/TX buffer to trigger interrupt.

The number of bytes in RX/TX buffer can be read from address 0x9c.

Shared under NDA

# 8 PWM

The TLSR8367 supports 5-channel PWM (Pulse-Width-Modulation) output. Each PWM#n (n=0~4) has its corresponding inverted output at PWM#n_N pin.

## 8.1 Register table

Table 1 Register table for PWM

| Address | Mnemonic | Type | Description | Reset Value |
|---|---|---|---|---|
| 0x780 | PWM_EN0 | R/W | [0]: Rsvd<br>[1:4]: Enable/Disable PWM1~PWM4<br>0--disable, 1--enable | 0x00 |
| 0x781 | PWM_EN1 | R/W | [0]: Enable/Disable PWM0<br>0--disable, 1--enable | 0x00 |
| 0x782 | PWM_CLK | R/W | (PWM_CLK+1)*sys_clk | 0x00 |
| 0x783 | PWM_MODE | R/W | [3:0]: PWM0 mode select<br>0000-pwm0 normal mode<br>0001-pwm0 count mode<br>0011-pwm0 IR mode<br>0111-pwm0 IR FIFO mode<br>1111-pwm0 IR DMA FIFO mode | 0x00 |
| 0x784 | PWM_CC0 | R/W | [4:0]:1'b1 invert PWM4~PWM0 output | 0x00 |
| 0x785 | PWM_CC1 | R/W | [4:0]:1'b1 invert PWM4_INV~ PWM0_INV output | 0x00 |
| 0x786 | PWM_CC2 | R/W | [4:0]: Signal frame polarity of PWM4~PWM0<br>1b'0-high level first<br>1b'1-low level first | 0x00 |
| 0x787 | PWM_32K_MODE | R/W | [0:4]: Set clock mode for PWM0~PWM4<br>1-- use pwm_32kclk<br>0-- use pwm_sysclk | 0x00 |
| 0x788~ 0x793 | reserved | | | |
| 0x794 | PWM_TCMP0 | R/W | [7:0] bits 7-0 of PWM0's high time or low time(if pola[0]=1) | 0x00 |
| 0x795 | PWM_TCMP0 | R/W | [15:8] bits 15-8 of PWM0's high time or low time | 0x00 |
| 0x796 | PWM_TMAX0 | R/W | [7:0] bits 7-0 of PWM0's cycle time | 0x00 |
| 0x797 | PWM_TMAX0 | R/W | [15:8] bits 15-8 of PWM0's cycle time | 0x00 |
| 0x798 | PWM_TCMP1 | R/W | [7:0] bits 7-0 of PWM1's high time or low time(if pola[1]=1) | 0x00 |
| 0x799 | PWM_TCMP1 | R/W | [15:8] bits 15-8 of PWM1's high time or low time | 0x00 |

Shared under NDA

| Address | Mnemonic | Type | Description | Reset Value |
|---|---|---|---|---|
| 0x79a | PWM_TMAX1 | R/W | [7:0] bits 7-0 of PWM1's cycle time | 0x00 |
| 0x79b | PWM_TMAX1 | R/W | [15:8] bits 15-8 of PWM1's cycle time | 0x00 |
| 0x79c | PWM_TCMP2 | R/W | [7:0] bits 7-0 of PWM2's high time or low time (if pola[2]=1) | 0x00 |
| 0x79d | PWM_TCMP2 | R/W | [15:8] bits 15-8 of PWM2's high time or low time | 0x00 |
| 0x79e | PWM_TMAX2 | R/W | [7:0] bits 7-0 of PWM2's cycle time | 0x00 |
| 0x79f | PWM_TMAX2 | R/W | [15:8] bits 15-8 of PWM2's cycle time | 0x00 |
| 0x7a0 | PWM_TCMP3 | R/W | [7:0] bits 7-0 of PWM3's high time or low time (if pola[3]=1) | 0x00 |
| 0x7a1 | PWM_TCMP3 | R/W | [15:8] bits 15-8 of PWM3's high time or low time | 0x00 |
| 0x7a2 | PWM_TMAX3 | R/W | [7:0] bits 7-0 of PWM3's cycle time | 0x00 |
| 0x7a3 | PWM_TMAX3 | R/W | [15:8] bits 15-8 of PWM3's cycle time | 0x00 |
| 0x7a4 | PWM_TCMP4 | R/W | [7:0] bits 7-0 of PWM4's high time or low time (if pola[4]=1) | 0x00 |
| 0x7a5 | PWM_TCMP4 | R/W | [15:8] bits 15-8 of PWM4's high time or low time | 0x00 |
| 0x7a6 | PWM_TMAX4 | R/W | [7:0] bits 7-0 of PWM4's cycle time | 0x00 |
| 0x7a7 | PWM_TMAX4 | R/W | [15:8] bits 15-8 of PWM4's cycle time | 0x00 |
| 0x7ac | PWM_PNUM0 | R/W | [7:0]PWM0 Pulse number in count mode and IR mode | 0x00 |
| 0x7ad | PWM_PNUM0 | R/W | [13:8] | 0x00 |
| 0x7ae~ 0x7af | reserved | | | |
| 0x7b0 | PWM_MASK0 | R/W | INT mask<br>[0] PWM0 Pnum int<br>0: disable 1: Enable<br>[1] PWM0 ir dma fifo mode int<br>0: disable 1: Enable<br>[2:6] PWM0~PWM4 frame int<br>0: disable, 1: enable | 0x00 |
| 0x7b1 | PWM_INT0 | R/W | INT status, write 1 to clear<br>[0]: PWM0 pnum int (have sent PNUM pulses, PWM_NCNT==PWM_PNUM)<br>[1]: PWM0 ir dma fifo mode int (pnum int &fifo empty in ir dma fifo mode)<br>[2:6]: PWM0~PWM4 cycle done int (PWM_CNT==PWM_TMAX) | 0x00 |
| 0x7b2 | PWM_MASK1 | R/W | [0]: PWM0 fifo mode fifo cnt int | 0x00 |

Shared under NDA

| Address | Mnemonic | Type | Description | Reset Value |
|---|---|---|---|---|
| | | | mask<br>0: disable, 1: enable | |
| 0x7b3 | PWM_INT1 | R/W | INT status, write 1 to clear<br>[0]: fifo mode cnt int, when FIFO_NUM (0x7cd[3:0]) is less than FIFO_NUM_LVL (0x7cc[3:0]) | 0x00 |
| 0x7b4 | PWM_CNT0 | R | [7:0]PWM0 cnt value | 0x00 |
| 0x7b5 | PWM_CNT0 | | [15:8]PWM0 cnt value | 0x00 |
| 0x7b6 | PWM_CNT1 | R | [7:0]PWM1 cnt value | 0x00 |
| 0x7b7 | PWM_CNT1 | | [15:8]PWM1 cnt value | 0x00 |
| 0x7b8 | PWM_CNT2 | R | [7:0]PWM2 cnt value | 0x00 |
| 0x7b9 | PWM_CNT2 | | [15:8]PWM2 cnt value | 0x00 |
| 0x7ba | PWM_CNT3 | R | [7:0]PWM3 cnt value | 0x00 |
| 0x7bb | PWM_CNT3 | | [15:8]PWM3 cnt value | 0x00 |
| 0x7bc | PWM_CNT4 | R | [7:0]PWM4 cnt value | 0x00 |
| 0x7bd | PWM_CNT4 | | [15:8]PWM4 cnt value | 0x00 |
| 0x7c0 | PWM_NCNT0 | R | [7:0]PWM0 pluse_cnt value | 0x00 |
| 0x7c1 | PWM_NCNT0 | | [15:8]PWM0 pluse_cnt value | 0x00 |
| 0x7c4 | PWM_TCMP0_SHADOW | R/W | [7:0] bits 7-0 of PWM0's high time or low time(if pola[0]=1), if shadow bit(fifo data[14]) is 1'b1 in ir fifo mode or dma fifo mode | 0x55 |
| 0x7c5 | PWM_TCMP0_SHADOW | R/W | [15:8] bits 15-8 of PWM0's high time or low time, if shadow bit(fifo data[14]) is 1'b1 in ir fifo mode or dma fifo mode | 0x55 |
| 0x7c6 | PWM_TMAX0_SHADOW | R/W | [7:0] bits 7-0 of PWM0's cycle time, if shadow bit(fifo data[14]) is 1'b1 in ir fifo mode or dma fifo mode | 0x00 |
| 0x7c7 | PWM_TMAX0_SHADOW | R/W | [15:8] bits 15-8 of PWM0's cycle time, if shadow bit(fifo frame[14]) is 1'b1 in ir fifo mode or dma fifo mode | 0x00 |
| 0x7c8 | FIFO_DAT0_ENTRY | W | Use in ir fifo mode | |
| 0x7c9 | FIFO_DAT1_ENTRY | W | Use in ir fifo mode | |
| 0x7ca | FIFO_DAT2_ENTRY | W | Use in ir fifo mode | |
| 0x7cb | FIFO_DAT3_ENTRY | W | Use in ir fifo mode | |
| 0x7cc | FIFO_NUM_LVL | R/W | FIFO num int trigger level | 0x00 |
| 0x7cd | FIFO_SR | R | [3:0]:FIFO DATA NUM(byte)<br>[4]:FIFO EMPTY<br>[5]:FIFO FULL | 0x00 |
| 0x7ce | FIFO_CLR | W1 | [0]: write 1 to clear data in FIFO | 0x00 |

Shared under NDA

## 8.2 Enable PWM

Register PWM_EN0 (address 0x780[4:1]) and PWM_EN1 (address 0x781[0]) serve to enable PWM4~PWM0 respectively via writing “1” for the corresponding bits.

Shared under NDA

## 8.3 Set PWM clock

PWM clock derives from system clock. Register PWM_CLK (address 0x782) serves to set the frequency dividing factor for PWM clock. Formula below applies:

$$F_{PWM} = F_{System\ clock} / (PWM\_CLK+1)$$

## 8.4 PWM waveform, polarity and output inversion

Each PWM channel has independent counter and 2 status including “Count” and “Remaining”. Count and Remaining status form a signal frame.

### 8.4.1 Waveform of signal frame

When PWM#n is enabled, first PWM#n enters Count status and outputs High level signal by default. When PWM#n counter reaches cycles set in register PWM_TCMP#n (address 0x794~0x795, 0x798~0x799, 0x79c~0x79d, 0x7a0~0x7a1, 0x7a4~0x7a5) / PWM_TCMP0_SHADOW (0x7c4~0x7c5), PWM#n enters Remaining status and outputs Low level till PWM#n cycle time configured in register PWM_TMAX#n (address 0x796~0x797, 0x79a~0x79b, 0x79e~0x79f, 0x7a2~0x7a3, 0x7a6~0x7a7) / PWM_TMAX0_SHADOW (0x7c6~0x7c7) expires.

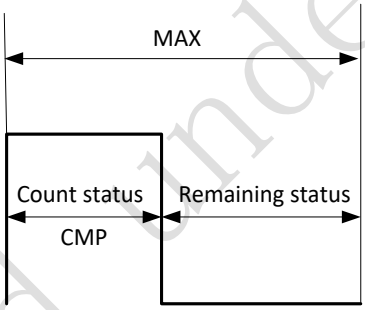



Figure 8- 1 A signal frame

An interruption will be generated at the end of each signal frame if enabled via register PWM_MASK0 (address 0x7b0[2:6]).

### 8.4.2 Invert PWM output

PWM#n and PWM#n_N output could be inverted independently via register PWM_CC0 (address 0x784) and PWM_CC1 (address 0x785). When the inversion bit is enabled, waveform of the corresponding PWM channel will be inverted completely.

### 8.4.3 Polarity for signal frame

By default, PWM#n outputs High level at Count status and Low level at Remaining status. When the corresponding polarity bit is enabled via register PWM_CC2 (address 0x786[4:0]), PWM#n will output Low level at Count status and High level at Remaining status.

Shared under NDA

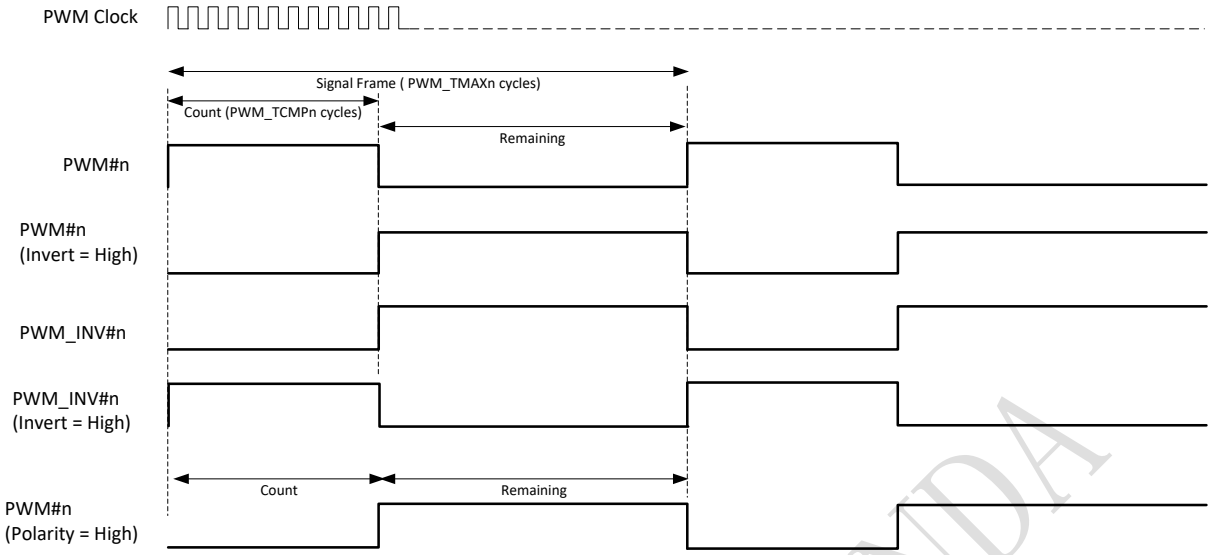



Figure 8- 2 PWM output waveform chart

## 8.5 PWM mode

### 8.5.1 Select PWM mode

PWM0 supports five modes, including Continuous mode (normal mode, default), Counting mode, IR mode, IR FIFO mode, IR DMA FIFO mode.

PWM1~PWM4 only support Continuous mode.

Register PWM_MODE (address 0x783) serves to select PWM0 mode.

### 8.5.2 Continuous mode

PWM0~PWM4 all support Continuous mode. In this mode, PWM#n continuously sends out signal frames. PWM#n should be disabled via address 0x780 or 0x781 to stop it; when stopped, the PWM output will turn low immediately.

During Continuous mode, waveform could be changed freely via PWM_TCMP#n and PWM_TMAX#n. New configuration for PWM_TCMP#n and PWM_TMAX#n will take effect in the next signal frame.

After each signal frame is finished, corresponding PWM cycle done interrupt flag bit (0x7b1[2:6]) will be automatically set to 1b'1. If the interrupt is enabled by setting PWM_MASK0 (address 0x7b0[2:6]) as 1b'1, a frame interruption will be generated. User needs to write 1b'1 to the flag bit to manually clear it.

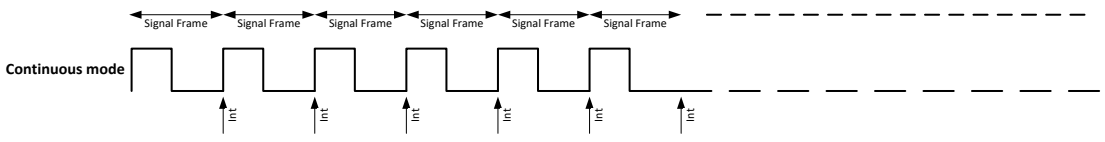



Figure 8-3 Continuous mode

Shared under NDA

#### 8.5.2.1 Set PWM clock mode

In continuous mode, by default PWM0~PWM4 use the system clock mode. To enable PWM to generate waveforms in suspend state, the 32kHz clock mode can be enabled via the configuration sequence below.

1) Disable PWM clock (0x64[4]=0), and enable PWM reset (0x61[4]=1).
2) Set 0x787[0]~[4] as 1b'1 to switch base clock to 32kHz for PWM0~PWM4.
3) Enable PWM clock (0x64[4]=1) and PWM_32kHz clock (0x65[1]=1), disable PWM reset (0x61[4]=0).

   Then corresponding PWM can work normally even in 32kHz clock.

### 8.5.3 Counting mode

Only PWM0 supports Counting mode. Address 0x783[3:0] should be set as 4b'0001 to select PWM0 counting mode.

In this mode, PWM0 sends out specified number of signal frames which is defined as a pulse group. The number is configured via register PWM_PNUM0 (address 0x7ac~0x7ad).

After each signal frame is finished, PWM0 cycle done interrupt flag bit (0x7b1[2]) will be automatically set to 1b'1. If the interrupt is enabled by setting PWM_MASK0 (address 0x7b0[2]) as 1b'1, a frame interruption will be generated. User needs to write 1b'1 to the flag bit to manually clear it.

After a pulse group is finished, PWM0 will be disabled automatically, and PWM0 pnum interrupt flag bit (0x7b1[0]) will be automatically set to 1b'1. If the interrupt is enabled by setting PWM_MASK0 (address 0x7b0[0]) as 1b'1, a Pnum interruption will be generated. User needs to write 1b'1 to the flag bit to manually clear it.

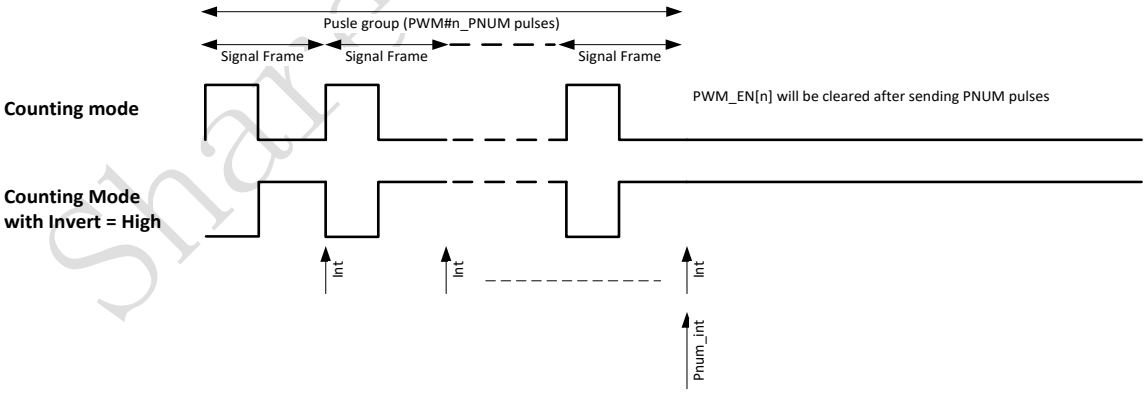



Figure 8-4 Counting mode (n=0)

Counting mode also serves to stop IR mode gracefully. Refer to Section 8.5.4 for details.

Shared under NDA

### 8.5.4 IR mode

Only PWM0 supports IR mode. Address 0x783[3:0] should be set as 4b'0011 to select PWM0 IR mode.

In this mode, specified number of frames is defined as one pulse group. In contrast to Counting mode where PWM0 stops after first pulse group is finished, PWM0 will constantly send pulse groups in IR mode.

During IR mode, PWM0 output waveform could also be changed freely via WM_TCMP0, PWM_TMAX0 and PWM_PNUM0. New configuration for PWM_TCMP0, PWM_TMAX0 and PWM_PNUM0 will take effect in the next pulse group.

To stop IR mode and complete current pulse group, user can switch PWM0 from IR mode to Counting mode so that PWM0 will stop after current pulse group is finished. If PWM0 is disabled directly via PWM_EN1 (0x781[0]), PWM0 output will turn Low immediately despite of current pulse group.

After each signal frame/pulse group is finished, PWM0 cycle done interrupt flag bit (0x7b1[2])/PWM0 pnum interrupt flag bit (0x7b1[0]) will be automatically set to 1b'1. A frame interruption/Pnum interruption will be generated (if enabled by setting address 0x7b0[2]/0x7b0[0] as 1b'1).

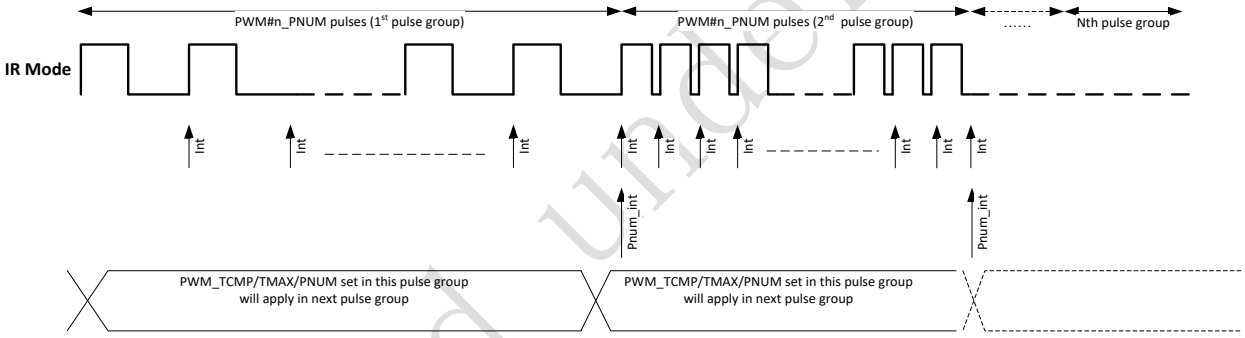



Figure 8-5 IR mode (n=0)

### 8.5.5 IR FIFO mode

IR FIFO mode is designed to allow IR transmission of long code patterns without the continued intervention of MCU, and it is designed as a selectable working mode on PWM0. The IR carrier frequency is divided down from the system clock and can be configured as any normal IR frequencies, e.g. 36kHz, 38kHz, 40kHz, or 56kHz.

Only PWM0 supports IR FIFO mode. Address 0x783[3:0] should be set as 4b'0111 to select PWM0 IR FIFO mode.

An element ("FIFO CFG Data") is defined as basic unit of IR waveform, and written into FIFO. This element consists of 16 bits, including:

- ✧ bit[13:0] defines PWM pulse number of current group.
- ✧ bit[14] determines duty cycle and period for current PWM pulse group.

  0: use configuration of TCMP0 and TMAX0 in 0x794~0x797;

  1: use configuration of TCMP0_SHADOW and TMAX0_SHADOW in 0x7c4~0x7c7.

Shared under NDA

✧ bit[15] determines whether current PWM pulse group is used as carrier, i.e. whether PWM will output pulse (1) or low level (0).

User should use FIFO_DATA_ENTRY in 0x7c8~0x7cb to write the 16-bit "FIFO CFG Data" into FIFO by byte or half word or word.

✧ To write by byte, user should successively write 0x7c8, 0x7c9, 0x7ca and 0x7cb.

✧ To write by half word, user should successively write 0x7c8 and 0x7ca.

✧ To write by word, user should write 0x7c8.

FIFO depth is 8 bytes. User can read the register FIFO_SR in 0x7cd to view FIFO empty/full status and check FIFO data number.

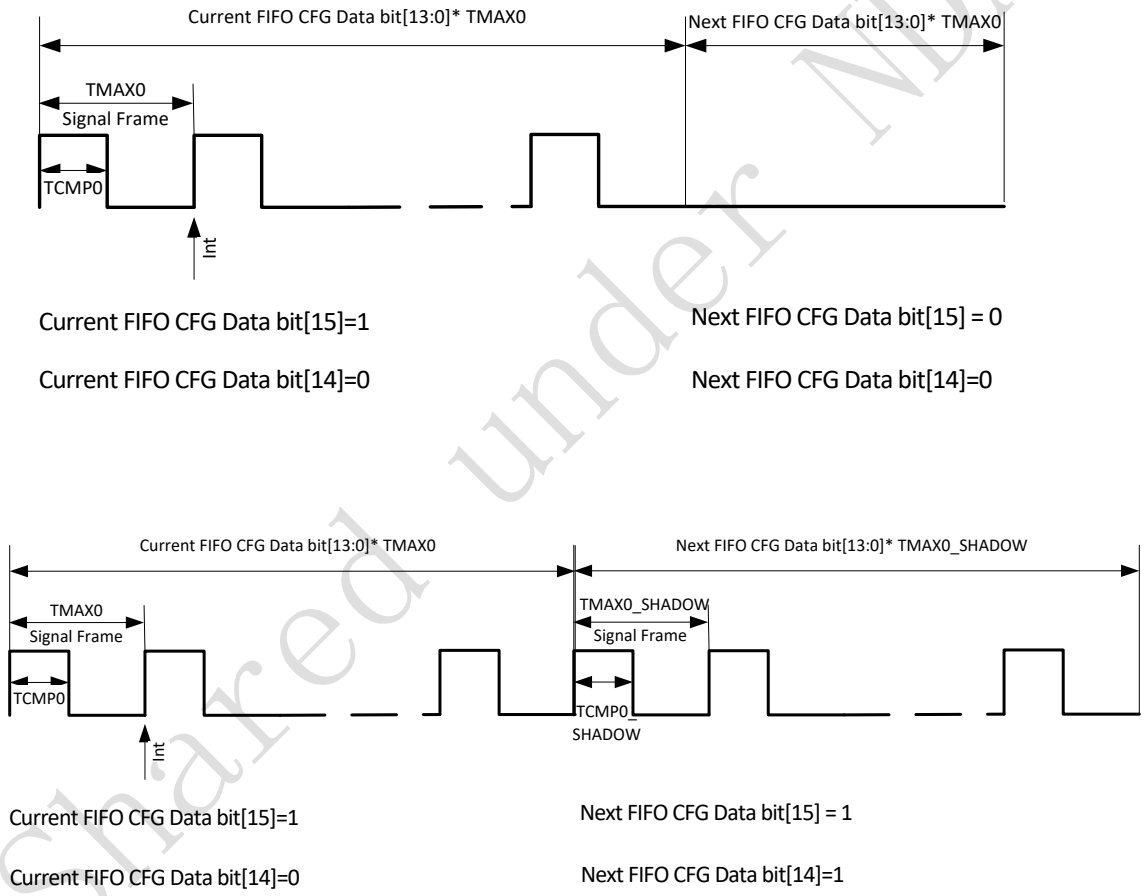



Figure 8- 6 IR format examples

When "FIFO CFG Data" is configured in FIFO and PWM0 is enabled via PWM_EN1 (address 0x781[0]), the configured waveforms will be output from PWM0 in sequence. As long as FIFO doesn't overflow, user can continue to add waveforms during IR waveforms sending process, and long IR code that exceeds the FIFO depth can be implemented this way. After all waveforms are sent, FIFO becomes empty, PWM0 will be disabled automatically.

Shared under NDA

The FIFO_CLR register (address 0x7ce[0]) serves to clear data in FIFO. Writing 1b'1 to this register will clear all data in the FIFO. Note that the FIFO can only be cleared when not in active transmission.

### 8.5.6 IR DMA FIFO mode

IR DMA FIFO mode is designed to allow IR transmission of long code patterns without occupation of MCU, and it is designed as a selectable working mode on PWM0. The IR carrier frequency is divided down from the system clock and can be configured as any normal IR frequencies, e.g. 36kHz, 38kHz, 40kHz, or 56kHz.

Only PWM0 supports IR DMA FIFO mode. Address 0x783[3:0] should be set as 4b'1111 to select PWM0 IR DMA FIFO mode.

This mode is similar to IR FIFO mode, except that "FIFO CFG Data" is written into FIFO by DMA instead of MCU. User should write the configuration of "FIFO CFG Data" into RAM, and then enable DMA channel 5. DMA will automatically write the configuration into FIFO.

***Note:** In this mode, when DMA channel 5 is enabled, PWM will automatically output configured waveform, without the need to manually enable PWM0 via 0x781[0] (i.e. 0x781[0] will be set as 1b'1 automatically).

**Example 1:**

**Suppose** Mark carrier (pulse) frequency1(F1) = 40kHz, duty cycle 1/3

Mark carrier (pulse) frequency2(F2) = 50kHz, duty cycle 1/2

Space carrier (low level) frequency(F3) = 40kHz

If user wants to make PWM send waveforms in following format (PWM CLK =24MHz):

Burst(20[F1]), i.e. 20 F1 pulses

Burst(30[F2]),

Burst(50[F1]) ,

Burst(50[F2]),

Burst(20[F1],10[F3]),

Burst(30[F2],10[F3])

**Step1:** Set carrier F1 frequency as 40kHz, set duty cycle as 1/3.

Set **PWM_TMAX0** as 0x258 (i.e. 24MHz/40kHz=600=0x258).

Since duty cycle is 1/3, set **PWM_TCMP0** as 0xc8 (i.e. 600/3=200=0xc8).

Set carrier F2 frequency as 50kHz, set duty cycle as 1/2.

Set **PWM_TMAX0_SHADOW** as 0x1e0 (i.e. 24MHz/50kHz=480=0x1e0).

Since duty cycle is 1/2, set **PWM_TCMP0_SHADOW** as 0xf0 (i.e. 480/2=240=0xf0).

Shared under NDA

**Step2:** Generate “FIFO CFG Data” sequence.

Burst(20[F1]): {[15]: 1’b1, [14]: 1’b0, [13:0]: ’d20}=0x8014.

Burst(30[F2]): {[15]: 1’b1, [14]: 1’b1, [13:0]: ’d30}=0xc01e.

Burst(50[F1]) : {[15]: 1’b1, [14]: 1’b0, [13:0]: ’d50}=0x8032.

Burst(50[F2]): {[15]: 1’b1, [14]: 1’b1, [13:0]:’d50}=0xc032.

Burst(20[F1],10[F3]): {[15]: 1’b1, [14]: 1’b0, [13:0]: ’d20}=0x8014,

{[15]: 1’b0, [14]: 1’b0, [13:0]: ’d10}=0x000a.

Burst(30[F2],10[F3]): {[15]: 1’b1, [14]: 1’b1, [13:0]: ’d30}=0xc01e,

{[15]:1’b0, [14]: 1’b0, [13:0]: ’d10}=0x000a.

**Step3:** Write “FIFO CFG Data” into SRAM in DMA format.

DMA SOURCE ADDRESS+0x00: 0x0000_0010 (dma transfer-length: 16byte)

DMA SOURCE ADDRESS+0x04: 0xc01e_8014 (LITTLE ENDIAN)

DMA SOURCE ADDRESS+0x08: 0xc032_8032

DMA SOURCE ADDRESS+0x0c: 0x000a_8014

DMA SOURCE ADDRESS+0x10: 0x000a_c01e

**Step4:** Enable DMA channel 5 to send PWM waveforms.

Write 1’b1 to address 0x524[5] to enable DMA channel 5.

After all waveforms are sent, FIFO becomes empty, PWM0 will be disabled automatically (address 0x781[0] is automatically cleared). The FIFO mode stop interrupt flag bit (address 0x7b3[0]) will be automatically set as 1b’1. If the interrupt is enabled by setting PWM_MASK1 (address 0x7b2[0]) as 1b’1, a FIFO mode stop interrupt will be generated. User needs to write 1b’1 to the flag bit to manually clear it.

**Example 2:**

**Suppose** carrier frequency is 38kHz, system clock frequency is 24MHz, duty cycle is 1/3, and the format of IR code to be sent is shown as below:

1) Preamble waveform: 9ms carrier + 4.5ms low level.
2) Data 1 waveform: 0.56ms carrier + 0.56ms low level.
3) Data 0 waveform: 0.56ms carrier + 1.69ms low level.
4) Repeat waveform: 9ms carrier + 2.25ms low level + 0.56ms carrier. Repeat waveform duration is 11.81ms, interval between two adjacent repeat waveforms is 108ms.
5) End waveform: 0.56ms carrier.

Shared under NDA

User can follow the steps below to configure related registers:

**Step1:** Set carrier frequency as 38kHz, set duty cycle as 1/3.

Set **PWM_TMAX0** as 0x277 (i.e. 24MHz/38kHz=631=0x277).

Since duty cycle is 1/3, set **PWM_TCMP0** as 0xd2 (i.e. 631/3=210=0xd2).

**Step2:** Generate "FIFO CFG Data" sequence.

**Preamble waveform:**

9ms carrier: {[15]:1'b1, [14]:1'b0, [13:0]: 9*38='d 342=14'h 156}=0x8156

4.5ms low level: {[15]:1'b0, [14]:1'b0, [13:0]: 4.5*38='d 171=14'h ab}=0x00ab

**Data 1 waveform:**

0.56ms carrier: {[15]:1'b1, [14]:1'b0, [13:0]: 0.56*38='d 21=14'h 15}=0x8015

0.56ms low level: {[15]:1'b0, [14]:1'b0, [13:0]: 0.56*38='d 21=14'h 15}=0x0015

**Data 0 waveform:**

0.56ms carrier: {[15]:1'b1, [14]:1'b0, [13:0]: 0.56*38='d 21=14'h 15}=0x8015

1.69ms low level: {[15]:1'b0, [14]:1'b0, [13:0]: 1.69*38='d 64=14'h 40}=0x0040

**Repeat waveform:**

9ms carrier: {[15]:1'b1, [14]:1'b0, [13:0]: 9*38='d 342=14'h 156}=0x8156

2.25ms low level: {[15]:1'b0, [14]:1'b0, [13:0]: 2.25*38='d 86=14'h 56}=0x0056

0.56ms carrier: {[15]:1'b1, [14]:1'b0, [13:0]: 0.56*38='d 21=14'h 15}=0x8015

108ms -11.81ms =96.19ms low level:

{[15]:1'b0, [14]:1'b0, [13:0]: 96.19*38='d 3655=14'h e47}=0x0e47

**End waveform:**

0.56ms carrier: {[15]:1'b1, [14]:1'b0, [13:0]: 0.56*38='d 21=14'h 15}=0x8015

**Step3:** Write "IR CFG Data" into SRAM in DMA format.

If user want PWM0 to send IR waveform in following format:

Preamble+0x5a+Repeat+End

Preamble: 0x8156, 0x00ab

0x5a=8'b01011010

Data 0: 0x8015, 0x0040

Data 1: 0x8015, 0x0015

Data 0: 0x8015, 0x0040

Data 1: 0x8015, 0x0015

Shared under NDA

Data 1: 0x8015, 0x0015

Data 0: 0x8015, 0x0040

Data 1: 0x8015, 0x0015

Data 0: 0x8015, 0x0040

Repeat: 0x8156, 0x0056, 0x8015, 0x0e47

End: 0x8015.

User needs to write the configuration information above into source address of DMA channel 5, as shown below:

DMA SOURCE ADDRESS+0x00: 0x0000_002e (dma transfer-length: 46byte)

DMA SOURCE ADDRESS+0x04: 0x00ab_8156 (Preamble) (LITTLE ENDIAN)

DMA SOURCE ADDRESS+0x08: 0x0040_8015 (Data 0)

DMA SOURCE ADDRESS+0x0c: 0x0015_8015 (Data 1)

DMA SOURCE ADDRESS+0x10: 0x0040_8015 (Data 0)

DMA SOURCE ADDRESS+0x14: 0x0015_8015 (Data 1)

DMA SOURCE ADDRESS+0x18: 0x0015_8015 (Data 1)

DMA SOURCE ADDRESS+0x1c: 0x0040_8015 (Data 0)

DMA SOURCE ADDRESS+0x20: 0x0015_8015 (Data 1)

DMA SOURCE ADDRESS+0x24: 0x0040_8015 (Data 0)

DMA SOURCE ADDRESS+0x28: 0x0056_8156 (Repeat)

DMA SOURCE ADDRESS+0x2c: 0x0e47_8015 (Repeat)

DMA SOURCE ADDRESS+0x30: 0x8015 (End)

**Step4:** Enable DMA channel 5 to send PWM waveforms.

Write 1'b1 to address 0x524[5] to enable DMA channel 5.

After all waveforms are sent, FIFO becomes empty, PWM0 will be disabled automatically (address 0x781[0] is automatically cleared). The FIFO mode stop interrupt flag bit (address 0x7b3[0]) will be automatically set as 1b'1. If the interrupt is enabled by setting PWM_MASK1 (address 0x7b2[0]) as 1b'1, a FIFO mode stop interrupt will be generated. User needs to write 1b'1 to the flag bit to manually clear it.

## 8.6 PWM interrupt

There are 8 interrupt sources from PWM function.

After each signal frame, PWM#n (n=0~4) will generate a frame-done IRQ (Interrupt Request) signal.

In Counting mode and IR mode, PWM0 will generate a Pnum IRQ signal after completing a pulse group.

shared under NDA

In IR FIFO mode, PWM0 will generate a FIFO mode count IRQ signal when the FIFO_NUM value is less than the FIFO_NUM_LVL, and will generate a FIFO mode stop IRQ signal after FIFO becomes empty.

In IR DMA FIFO mode, PWM0 will generate an IR waveform send done IRQ signal, after DMA has sent all configuration data, FIFO becomes empty and final waveform is sent.

To enable PWM interrupt, the total enabling bit “irq_pwm” (address 0x641[6], see **section 6 Interrupt**) should be set as 1b’1. To enable various PWM interrupt sources, PWM_MASK0 (address 0x7b0[6:0]) and PWM_MASK1 (address 0x7b2[0]) should be set as 1b’1 correspondingly.

Interrupt status can be cleared via register PWM_INT0 (address 0x7b1[6:0]) and PWM_INT1 (address 0x7b3[0]).

Shared under NDA

# 9 Quadrature Decoder

The TLSR8367 embeds one quadrature decoder (QDEC) which is designed mainly for applications such as wheel. The QDEC implements debounce function to filter out jitter on the two phase inputs, and generates smooth square waves for the two phase.

## 9.1 Input pin selection

The QDEC supports two-phase input; each input is selectable from the 8 pins of PortA, PortB, PortC and PortD via setting address 0xd2[2:0] (for channel a)/0xd3[2:0] (for channel b).

Table 9- 1 Input pin selection

| Address 0xd2[2:0]/0xd3[2:0] | Pin |
|---|---|
| 0 | PD[1] |
| 1 | PD[3] |
| 2 | PC[3] |
| 3 | PB[3] |
| 4 | PB[4] |
| 5 | PB[5] |
| 6 | PA[0] |
| 7 | PA[1] |

**Note:** To use corresponding IO as QDEC input pin, it's needed first to enable GPIO function, enable "IE" (1) and disable "OEN" (1) for this IO.

## 9.2 Common mode and double accuracy mode

The QDEC embeds an internal hardware counter, which is not connected with bus.

Address 0xd7[0] serves to select common mode or double accuracy mode.

For each wheel rolling step, two pulse edges (rising edge or falling edge) are generated.

If address 0xd7[0] is cleared to select common mode, the QDEC Counter value (real time counting value) is increased/decreased by 1 only when the same rising/falling edges are detected from the two phase signals.

Shared under NDA

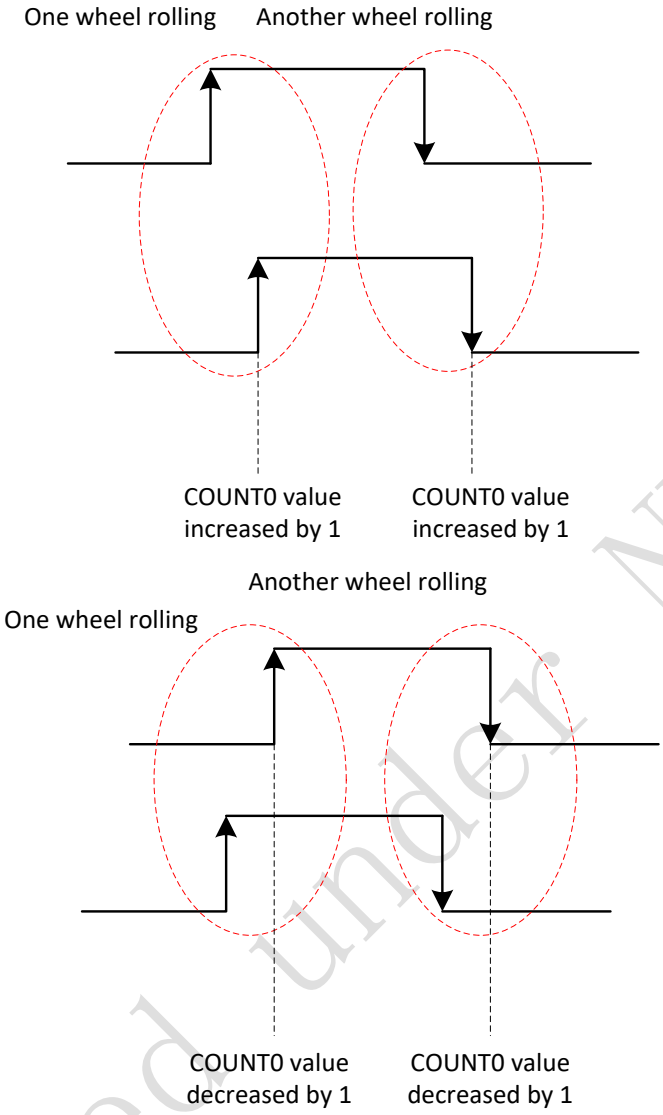



Figure 9- 1 Common mode

If address 0xd7[0] is set to 1b'1 to select double accuracy mode, the QDEC Counter value (real time counting value) is increased/decreased by 1 on each rising/falling edge of the two phase signals; the COUNT0 will be increased/decreased by 2 for one wheel rolling.

Shared under NDA

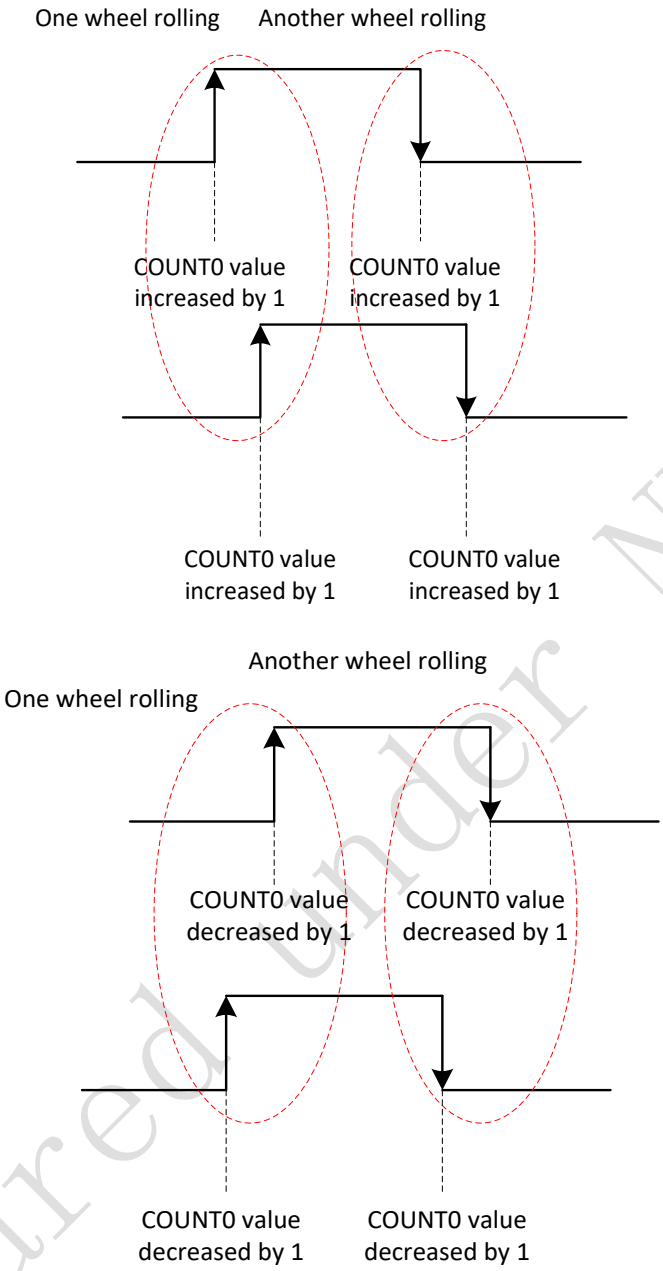



Figure 9- 2 Double accuracy mode

Shared under NDA

## 9.3 Read real time counting value

Neither can Hardware Counter value be read directly via software, nor can the counting value in address 0xd0 be updated automatically.

To read real time counting value, first write address 0xd8[0] with 1b'1 to load Hardware Counter data into the QDEC_COUNT register, then read address 0xd0.

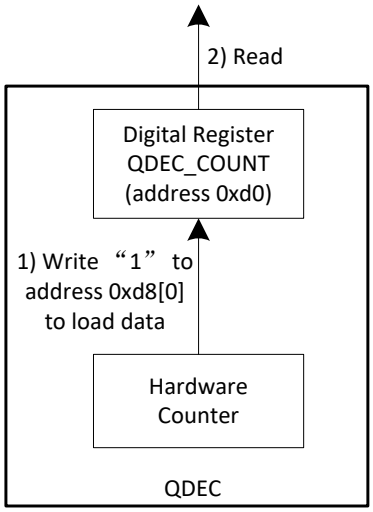



Figure 9- 3 Read real time counting value

## 9.4 QDEC reset

Address 0xd6[0] serves to reset the QDEC. The QDEC Counter value is cleared to zero.

## 9.5 Other configuration

The QDEC supports hardware debouncing. Address 0xd1[2:0] serves to set filtering window duration. All jitter with period less than the value will be filtered out and thus does not trigger count change.

Address 0xd1[4] serves to set input signal initial polarity.

Address 0xd1[5] serves to enable shuttle mode. Shuttle mode allows non-overlapping two phase signals as shown below.

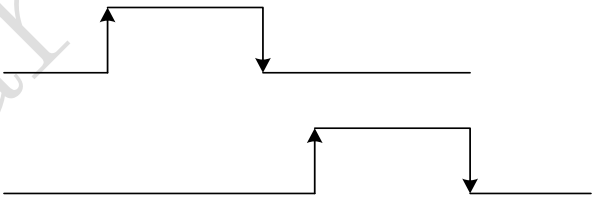


Figure 9- 4 Shuttle mode

Shared under NDA

## 9.6 Timing sequence

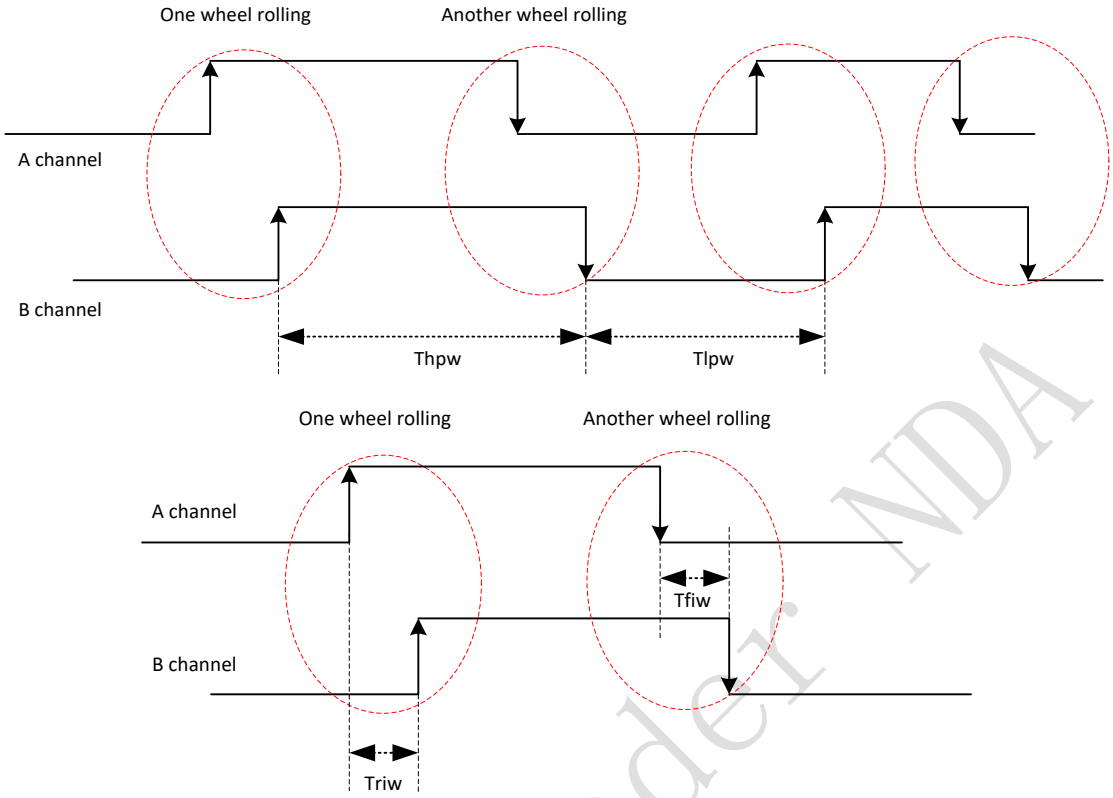



Figure 9- 5 Timing sequence chart

Table 9- 2 Timing

| Time interval | Min Value |
|---|---|
| Thpw (High-level pulse width) | 2^(n+1) *clk_32kHz *3 (n=0xd1[2:0]) |
| Tlpw (Low-level pulse width) | 2^(n+1) *clk_32kHz *3 (n=0xd1[2:0]) |
| Triw (Interval width between two rising edges) | 2^(n+1) *clk_32kHz (n=0xd1[2:0]) |
| Tfiw (Interval width between two falling edges) | 2^(n+1) *clk_32kHz (n=0xd1[2:0]) |

QDEC module works based on 32kHz clock to ensure it can work in suspend mode. QDEC module supports debouncing function, and any signal with width lower than the threshold (i.e. “2^(n+1) *clk_32kHz *3 (n=0xd1[2:0])) will be regarded as jitter. Therefore, effective signals input from Channel A and B should contain high/low level with width Thpw/Tlpw more than the threshold. The 2^n *clk_32kHz clock is used to synchronize input signal of QDEC module, so the interval between two adjacent rising/falling edges from Channel A and B, which are marked as Triw and Tfiw, should exceed “2^(n+1) *clk_32kHz”.

Only when the timing requirements above are met, can QDEC module recognize wheel rolling times correctly.

Shared under NDA

## 9.7 Register table

Table 9- 3 Register table for QDEC

| Address | Mnemonic | Type | Description | Reset value |
|---|---|---|---|---|
| 0xd0 | QDEC_COUNT | R | QDEC Counting value (read to clear):<br>Pulse edge number | 0x00 |
| 0xd1 | QDEC_CC | R/W | [2:0] :<br>filter time (can filter 2^n *clk_32kHz*2 width deglitch)<br>[4]: pola, input signal pola<br>0: no signal is low, 1: no signal is high<br>[5]: shuttle mode<br>1 to enable shuttle mode | 0x00 |
| 0xd2 | QDEC_CHNA | R/W | [2:0] QDEC input pin select for channel a<br>choose 1 of 8 pins for input channel a<br>0~7: {PD[1], PD[3], PC[3], PB[3]~PB[5], PA[0]~PA[1]} | 0x00 |
| 0xd3 | QDEC_CHNB | R/W | [2:0] QDEC input pin select for channel b choose 1 of 8 pins for input channel b<br>0~7: {PD[1], PD[3], PC[3], PB[3]~PB[5], PA[0]~PA[1]} | 0x01 |
| 0xd6 | QDEC_RST | R/W | [0]Write 1 to reset QDEC | 0x00 |
| 0xd7 | QDEC_DOUBLE | R/W | [0]QDEC mode select<br>0-Select common mode<br>1-Enable double accuracy mode | 0x00 |
| 0xd8 | DATA_LOAD | R/W | [0]write 1 to load data<br>when load completes it will be 0 | 0x00 |

Shared under NDA

# 10 SAR ADC

The TLSR8367 integrates one SAR ADC module, which can be used to sample analog input signals such as battery voltage and external analog input.

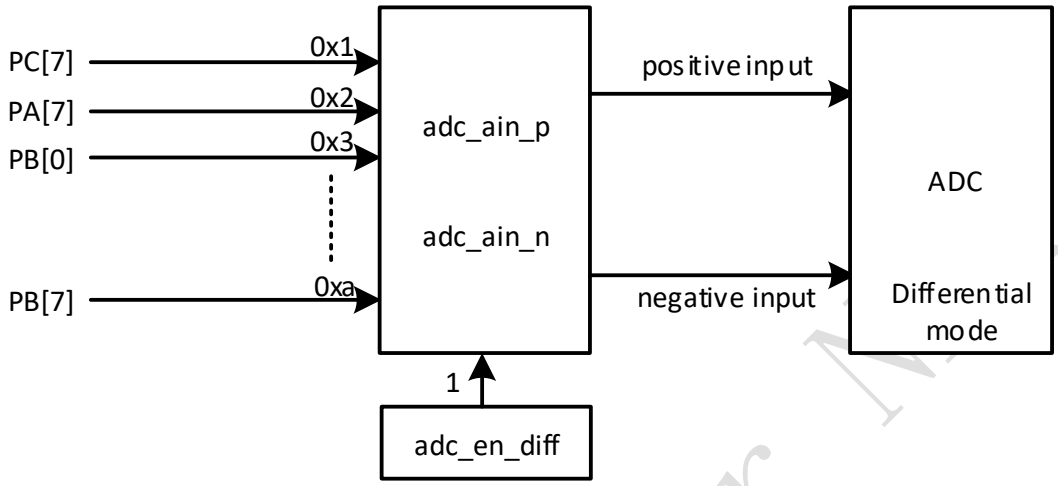



Figure 10- 1 Block diagram of ADC

## 10.1 Power on/down

The SAR ADC is disabled by default. To power on the ADC, the analog register adc_pd (afe_0xfc<5>) should be set as 1b'0.

## 10.2 ADC clock

ADC clock is derived from external 24MHz crystal source, with frequency dividing factor configurable via the analog register adc_clk_div (afe_0xf4<2:0>).

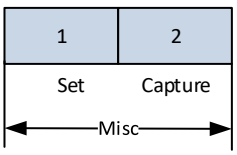
ADC clock frequency (marked as $F_{ADC\_clk}$) = 24MHz/(adc_clk_div+1)

## 10.3 ADC control in auto mode

### 10.3.1 Set max state and enable channel

The SAR ADC supports Misc channel. The Misc channel consists of one "Set" state and one "Capture" state.

✧ The analog register r_max_scnt (afe_0xf2<6:4>) serves to set the max state index. As shown below, the r_max_scnt should be set as 0x02.




Shared under N

- ✧ The Misc channel can be enabled via r_en_misc (afe_0xf2<2>).
- ✧ RSSI signal sampling channel is multiplexed with the Misc channel.

  Note: To sample RSSI signal, it's not needed to enable the Misc channel, and Misc channel input is automatically set as differential RSSI_n and RSSI_p. Both sampling time and resolution differ.

### 10.3.2 "Set" state

The length of "Set" state for Misc channel is configurable via the analog register r_max_s (afe_0xf1<3:0>).

"Set" state duration (marked as $T_{sd}$) = r_max_s / 24MHz.

"Set" state serves to set ADC control signals for Misc channel via corresponding analog registers, including:

- ✧ adc_en_diff: afe_0xec<6> (Misc channel). Must set as 1b'1 to select differential input mode.
- ✧ adc_ain_p: afe_0xe8<7:4> (Misc channel). Select positive input in differential mode.
- ✧ adc_ain_n: afe_0xe8<3:0> (Misc channel). Select negative input in differential mode.

  ***Note:** For RSSI signal sample channel, differential input mode is automatically selected, without the need to set afe_0xec<6>, while Misc channel input is automatically set as differential RSSI_n and RSSI_p without the need to set afe_0xe8.

- ✧ adc_vref: afe_0xe7<5:4> (Misc channel). Set reference voltage $V_{REF}$. ADC maximum input range is the determined by the ADC reference voltage.
- ✧ adc_sel_ai_scale: afe_0xfa<7:6>. Set scaling factor for ADC analog input as 1 (default), or 1/8.

  By setting this scaling factor, ADC maximum input range can be extended based on the $V_{REF}$.

For example, suppose the $V_{REF}$ is set as 1.2V:

Since the scaling factor is 1 by default, the ADC maximum input range should be 0~1.2V (negative input is GND) / -1.2V~+1.2V (negative input is ADC GPIO pin).

If the scaling factor is set as 1/8, ADC maximum input range should change to 0~9.6V (negative input is GND) / -9.6V~+9.6V (negative input is ADC GPIO pin).

Shared under NDA

✧ adc_res: afe_0xec<1:0> (Misc channel). Set resolution as 8/10/12/14 bits.

***Note:** For RSSI signal sample channel, resolution is fixed as 8bits without the need to set afe_0xec<1:0>.

ADC data is always 15-bit format no matter how the resolution is set. For example, 14 bits resolution indicates ADC data consists of 14-bit valid data and 1-bit sign extension bit.

✧ adc_tsamp: afe_0xee<3:0> (Misc channel), afe_0xee<7:4> (RSSI signal sample channel). Set sampling time which determines the speed to stabilize input signals.

Sampling time (marked as $T_{samp}$) = adc_tsamp / $F_{ADC\_clk}$.

The lower sampling cycle, the shorter ADC convert time.

✧ pga_boost, pga_gain: Set PGA gain in Boost stage and Gain stage. See PGA section.

### 10.3.3 "Capture" state

For the Misc channel, at the beginning of "Capture" state, run signal is issued automatically to start an ADC sampling and conversion process; at the end of "Capture" state, ADC output data is captured.

✧ The length of "Capture" state for Misc channel is configurable via the analog register r_max_mc[9:0] (afe_0xf1<7:6>, afe_0xef<7:0>).

"Capture" state duration for Misc channel (marked as $T_{cd}$) = r_max_mc / 24MHz.

✧ The "VLD" bit (afe_0xf8<7>) will be set as 1b'1 at the end of "Capture" state to indicate the ADC data is valid, and this flag bit will be cleared automatically.

✧ The 15-bit ADC output data for Misc channel can be read from the analog register adc_dat[14:0] (afe_0xf8<6:0>, afe_0xf7<7:0>).

Note: The total duration "$T_{td}$", which is the sum of the length of "Set" state and "Capture" state for Misc channel available, determines the sampling rate.

Sampling frequency (marked as $F_s$) = 1 / $T_{td}$

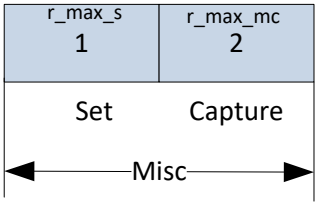
Shared under NDA

### 10.3.4 Usage cases

#### 10.3.4.1 Case 1: 1-channel sampling for Misc

In this case, afe_0xf2<3:0> should be set as 0x4, so as to enable the Misc channel, the max state index should be set as “2” by setting afe_0xf2 as 0x2.

The total duration (marked as $T_{td}$) = (1*r_max_s+1*r_max_mc) / 24MHz.




#### 10.3.4.2 Case 2: RSSI capture

RSSI signal sampling channel is shared with the Misc channel. It’s not needed to set r_en_misc (afe_0xf2<2>) to enable Misc channel, while sampling time and resolution differ.

In this case, the rst_st_en (afe_0xf4<7>) may be set as 1b’1, which means when RSSI (Received Signal Strength Indication) signal is ready for measurement, and the state machine will be restarted to ensure that the measurement starts from the “Set” state.

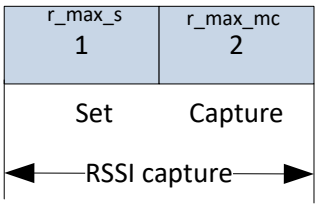
In the “Set” state, the adc_tsamprssi (afe_0xee<7:4>) should be set to configure sampling time.

The sampling resolution is automatically set as 8bits.

RSSI capture channel is automatically set as differential RSSI_n and RSSI_p input.

The other configurations are the same as the Misc channel.

The total duration (marked as $T_{td}$) = (1*r_max_s+1*r_max_mc) / 24MHz.




Shared under NDA

#### 10.3.4.3 Case 3 with detailed register setting

This case introduces the register setting details for Misc channel.

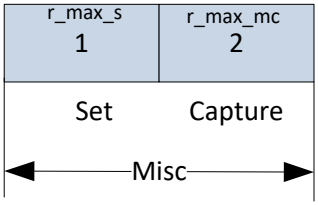



Table 10- 1 Overall register setting

| Function | Register setting |
|---|---|
| Power on the ADC | afe_0xfc<5> = 1b'0 |
| Set $F_{ADC\_clk}$ (ADC clock frequency) as 4MHz | afe_0xf4<2:0> = 5<br>$F_{ADC\_clk}$ = 24MHz/(5+1)=4MHz |
| Enable the Misc channel | afe_0xf2<3:0> = 0x4 |
| Set the max state index as "2" | afe_0xf2<6:4> = 0x2 |

Table 10- 2 Register setting for M channel

| Function | Register setting for Misc |
|---|---|
| Set $T_{sd}$ ("Set" state duration) | afe_0xf1<3:0> = 10<br>$T_{sd}$ = r_max_s/24MHz = 10/24MHz = 0.417us |
| Set $T_{cd}$ ("Capture" state duration) | afe_0xf1<7:6>=1, afe_0xef<7:0>=0xea<br>$T_{cd}$ = r_max_mc[9:0]/24MHz = 490/24MHz = 20.417us |
| $T_{td}$ (total duration) | $T_{td}$ = (1*r_max_s+1*r_max_mc) / 24MHz = 500/24MHz =20.83us |
| $F_s$ (Sampling frequency) | $F_s$ = 1 / $T_{td}$ = 24MHz/500 = 48kHz |
| Set differential input | afe_0xec<6>=1 |
| Set input channel | afe_0xe8=0xaf<br>Select PB[7] and GND as positive input and negative input |
| Set reference voltage $V_{REF}$ | afe_0xe7<5:4>=2<br>$V_{REF}$ =1.2V |
| Set scaling factor for ADC analog input | afe_0xfa<7:6>=0<br>scaling factor: 1 |
| | ADC maximum input range: 0 ~ +1.2V |
| Set resolution | afe_0xec<1:0>=3<br>resolution: 14bits |
| Set $T_{samp}$ (determines the speed to stabilize input before sampling) | afe_0xee<3:0>=3<br>$T_{samp}$ = adc_tsamp / $F_{ADC\_clk}$ = 12/4MHz=3us |

Shared under NDA

## 10.4 Register table

Table 10- 3 Register table related to SAR ADC

| Address | Mnemonic | Default value | Description |
|---|---|---|---|
| afe_0xe7<5:4> | adc_vrefm | 00 | Select $V_{REF}$ for Misc channel<br>0x0: rsvd<br>0x1: 0.9V<br>0x2: 1.2V<br>0x3: rsvd |
| afe_0xe8<3:0> | adc_ain_m_n | 0010 | Select negative input for Misc channel:<br>0x0: No input<br>0x1: PC[7]<br>0x2: PA[7]<br>0x3: PB[0]<br>0x4: PB[1]<br>……<br>0xa: PB[7]<br>0xb: pga_channel_n<0> (PGA negative output)<br>0xc: rsvd<br>0xd: rsvd<br>0xe: rsvd<br>0xf: Ground |
| afe_0xe8<7:4> | adc_ain_m_p | 0000 | Select positive input for Misc channel:<br>0x0: No input<br>0x1: PC[7]<br>0x2: PA[7]<br>0x3: PB[0]<br>0x4: PB[1]<br>……<br>0xa: PB[7]<br>0xb: pga_channel_p<0> (PGA positive output)<br>0xc: rsvd<br>0xd: rsvd<br>0xe: rsvd<br>0xf: rsvd |
| afe_0xec<1:0> | adc_resm | 11 | Set resolution for Misc channel<br>0x0: 8bits<br>0x1: 10bits<br>0x2: 12bits<br>0x3: 14bits |
| afe_0xec<6> | adc_en_diffm | 0 | Select input mode for Misc channel.<br>0: rsvd<br>1: differential mode |

Shared under NDA

| Address | Mnemonic | Default value | Description |
|---|---|---|---|
| afe_0xee<3:0> | adc_tsampm | 0001 | Number of ADC clock cycles in sampling phase for Misc channel to stabilize the input before sampling:<br>0x0: 3 cycles<br>0x1: 6 cycles<br>0x2: 9 cycles<br>0x3: 12 cycles<br>…<br>0xf: 48 cycles |
| afe_0xee<7:4> | adc_tsamprssi | 0000 | |
| afe_0xef<7:0> | r_max_mc[7:0] | 0x0f | r_max_mc[9:0]serves to set length of “capture” state for Misc channel.<br>r_max_s serves to set length of “set” state for Misc channel.<br>Note: State length indicates number of 24MHz clock cycles occupied by the state. |
| afe_0xf1<3:0> | r_max_s | 0110 | |
| afe_0xf1<7:6> | r_max_mc[9:8] | 00 | |
| afe_0xf2<2> | r_en_misc | 1 | Enable Misc channel sampling. 1: enable |
| afe_0xf2<3> | rsvd | 0 | rsvd |
| afe_0xf2<6:4> | r_max_scnt | 010 | Set total length for sampling state machine (i.e. max state index) |
| afe_0xf4<2:0> | adc_clk_div | 011 | ADC clock (derive from external 24MHz crystal)<br>ADC clock frequency = 24MHz/(adc_clk_div+1) |
| afe_0xf4<7> | rst_st_en | 0 | 1: enable state machine restart, 0: disable |
| afe_0xf5<7:0> | rsvd | 0x10 | rsvd |
| afe_0xf6<7:0> | rsvd | 0x11 | rsvd |
| afe_0xf7<7:0> | adc_dat[7:0] | 0x00 | Read only, Misc adc dat[7:0] |
| afe_0xf8<7:0> | adc_dat[15:8] | 0x00 | Read only<br>[7]: vld, ADC data valid status bit (This bit will be set as 1 at the end of capture state to indicate the ADC data is valid, and will be cleared when set state starts.)<br>[6:0]: Misc adc_dat[14:8] |
| afe_0xf9<3:2> | rsvd | 00 | rsvd |
| afe_0xf9<5> | | 0 | Must be set as 1b’1 |
| afe_0xfa<7:6> | adc_sel_ai_scale | 0 | Analog input pre-scaling select<br>sel_ai_scale[1:0]: scaling factor<br>0x0: 1<br>0x1: rsvd<br>0x2: rsvd<br>0x3: 1/8 |
| afe_0xfc<4> | rsvd | 0 | rsvd |
| afe_0xfc<5> | adc_pd | 1 | Power down ADC<br>1: Power down<br>0: Power up |

Shared under NDA

# 11 PGA

The TLSR8367 integrates a PGA (Programmable Gain Amplifier) module.

The PGA consists of Boost stage pre-amplifier and Gain stage post-amplifier.

The PGA is used in combination with the ADC module: By adjusting the gain of pre-amplifier and post-amplifier, the PGA can amplify differential analog input signals from specific pins before ADC sampling.

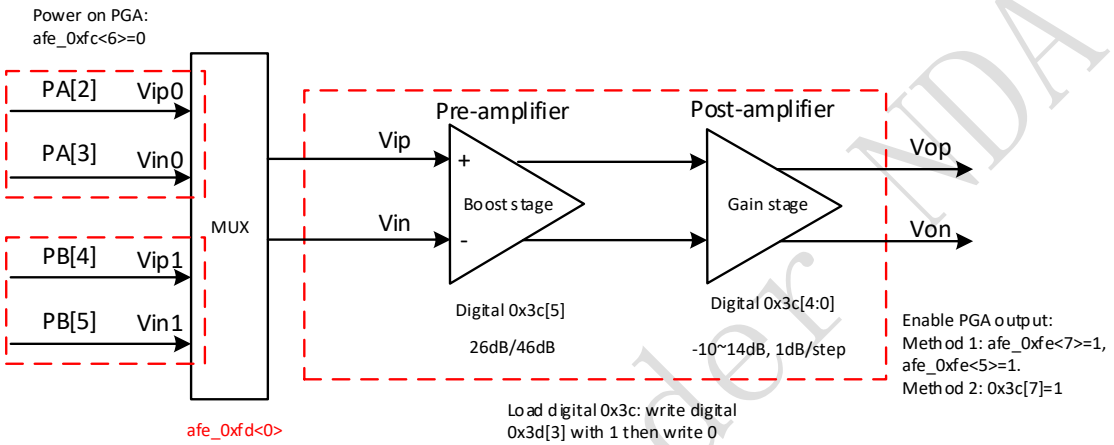



Figure 11- 1 Block diagram of PGA

***Note:**

Vip<0>, Vin<0>: Positive input 0 and Negative input 0;

Vip<1>, Vin<1>: Positive input 1 and Negative input 1;

Vop, Von: Positive and Negative output.

Shared under NDA

### 11.1 Power on/down

The PGA is disabled by default.

To power on the PGA, the analog register pga_pd (afe_0xfc<6>) should be set as 1b'0.

### 11.2 Select input channel

The analog register pga_sel_vin (afe_0xfd<0>) should be set as 1b'1 to select PB[4] and PB[5] as positive and negative input, respectively.

### 11.3 Adjust gain

The PGA gain is adjustable via digital register 0x3c:

- ✧ Address 0x3c[5] serves to set gain for the pre-amplifier as 26dB or 46dB.
- ✧ Address 0x3c[4:0] serves to set gain for the post-amplifier as -10dB (0x0) ~ 14dB (0x18) with step of 1dB.
- ✧ The total PGA gain should be the sum of the two gain values.

### 11.4 Enable/Disable PGA output

User can enable/disable PGA output by using either analog registers or digital register.

- ✧ Use analog register "pga_mute_m_en" (afe_0xfe<7>) and "pga_mute_m" (afe_0xfe<5>). User can enable PGA output by setting both afe_0xfe<7> and afe_0xfe<5> as 1b'1.
- ✧ Use digital register 0x3c[7]. User can also enable PGA output by setting 0x3c[7] as 1b'1.
- ✧ Digital register 0x3d[4] indicates whether PGA output is enabled or disabled.

### 11.5 Load digital register 0x3c

User should write digital register 0x3d[3] with 1b'1 and 1b'0 to load the value of digital register 0x3c, so that the configuration of 0x3c can take effect.

Shared under NDA

## 11.6 Register table

Table 11- 1 Analog register table related to PGA

| Address | Mnemonic | Default | Description |
|---|---|---|---|
| **Analog register** | | | |
| afe_0xfc<6> | pga_pd | 1 | Power down PGA<br>1: Power down, 0: Power up |
| afe_0xfd<0> | pga_sel_vin | 0 | Select PGA differential input source. Gate off all input with pga_pd.<br>0: select PA[2] (positive) and PA[3] (negative)<br>1: select PB[4] (positive) and PB[5] (negative) |
| afe_0xfe<7> | pga_mute_m_en | 0 | Enable using analog register pga_mute_m<br>0: disable using analog register afe_0xfe<5><br>1: enable using analog register afe_0xfe<5> |
| afe_0xfe<5> | pga_mute_m | 0 | When afe_0xfe<7> = 0x1, this bit can be used<br>0: not mute pga<br>1: mute pga |
| **Digital register** | | | |
| 0x3c | | 0x00 | [4:0]: set gain for the post-amplifier<br>0x0~0x18: -10dB~14dB, step 1dB<br>[5]: set gain for the pre-amplifier<br>0: 26dB, 1: 46dB<br>[7]: mute PGA using digital register |
| 0x3d | | 0x00 | [3]: write 1 and then write 0 to trigger PGA load 0x3c value<br>[4]: read only indicate PGA mute release status<br>0: not release<br>1: release |

Shared under NDA

# 12 EEPROM

The TLSR8367E02 embeds 2Kbit EEPROM. To control the EEPROM, PA[5]/PB[6] inside the TLSR8367E02 are connected to the EEPROM and serve as I2C SCL/SDA respectively.

## 12.1 Communication protocol

Both SDA and SCL are bidirectional lines connected to a positive supply voltage via a pull-up resister. When the bus is free, both lines are HIGH. Data in SDA line must keep stable when SCL line is at high level, and it's only allowed to change when SCL line is at low level.

A negative/positive edge of SDA when SCL is high indicate a start/stop condition, respectively.

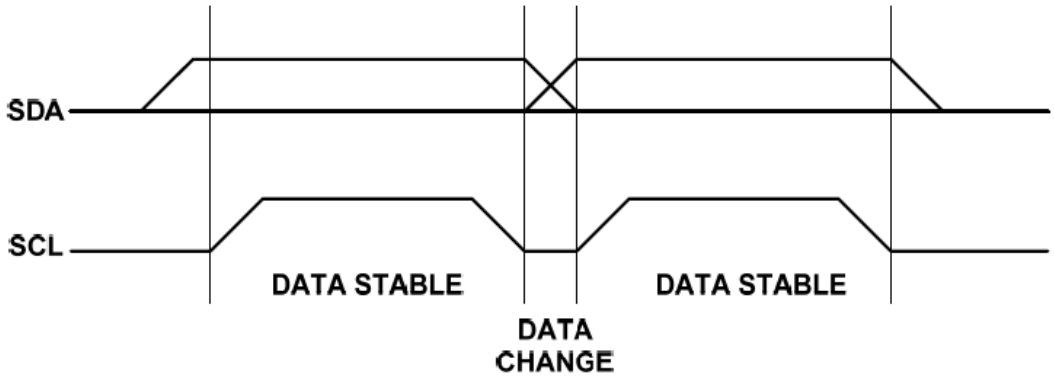



Figure 12- 1 Data validity

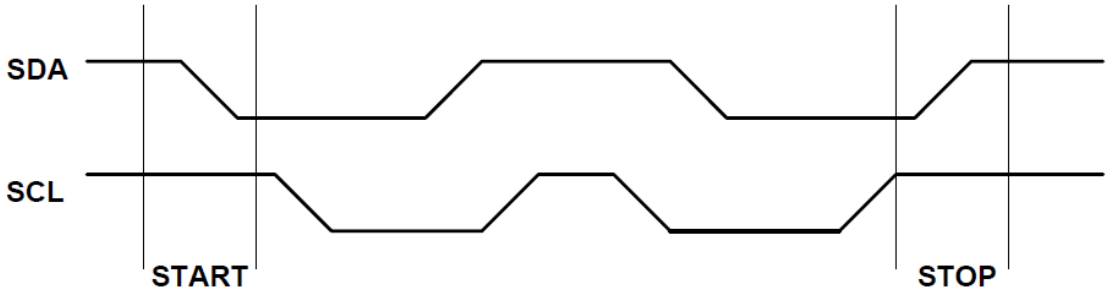



Figure 12- 2 Start and stop condition

All addresses and data words are serially transferred between the EEPROM and the MCU in 8-bit words. The EEPROM will respond with an ack “0” after it receives each word. Upon receipt of each word from the EEPROM, the MCU should also send a “0” to the EEPROM, and continue to output the next data word or send a stop condition.

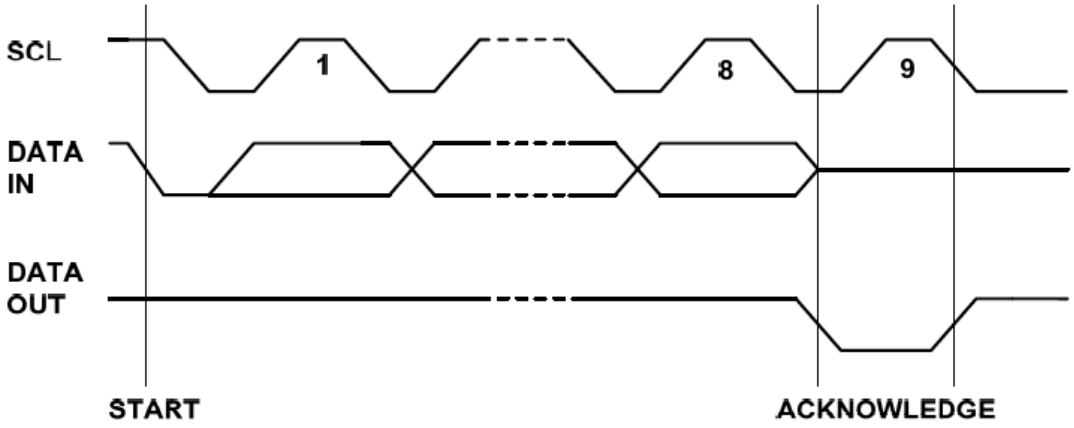



Figure 12- 3 Send Ack

Upon power-up or receipt of the stop bit and completion of any internal operations, the EEPROM will enter low-power standby mode.

The EEPROM requires an 8-bit device address word following a start condition to enable the read/write access operation. As shown in Figure 12- 4, the device address word consists of a mandatory 1, 0 sequence for the first four MSBs, device address bits A2, A1 and A0, as well as R(1)/W(0) select bit to indicate read/write operation.

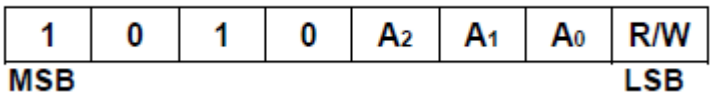



Figure 12- 4 Device address

Since the device address bits “A2A1A0” are “000”, EEPROM I2C address should be 0x50.

Shared under

## 12.2 EEPROM operation

For EEPROM read and write operations, user needs to simulate corresponding I2C read/write timing sequence via software.

### 12.2.1 Write operations

The EEPROM supports byte write and 8-byte page write.

For byte write, a write operation requires an 8-bit data word address following the device address word and ack. Upon receipt of this address, the EEPROM will also respond with an ack “0”, and then the first 8-bit data word is clocked in. After the 8-bit data word is received, the EEPROM will send an ack “0”. The addressing device (MCU) must terminate the write sequence with a stop condition. At this time, the EEPROM enters an internally-timed write cycle to the nonvolatile memory. All inputs are disabled during this write cycle, and the EEPROM won’t respond until the write is completed.

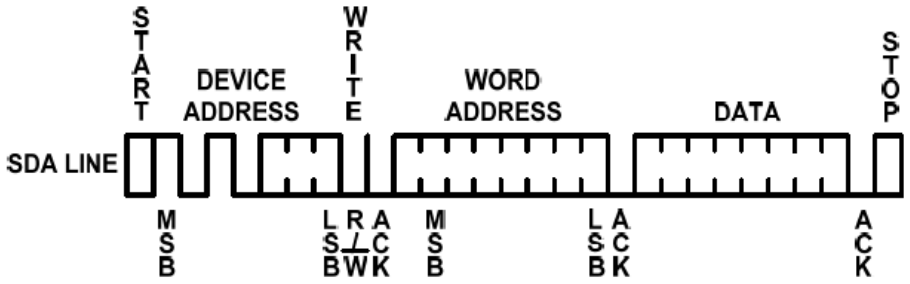



Figure 12- 5 Byte write

A page write initiation is the same as a byte write. The MCU does not send a stop condition after the first data word is clocked in and acknowledged by the EEPROM, and up to seven more data words can be transmitted. The EEPROM will respond with an ack “0” after each data word is received. The MCU must terminate the page write with a stop condition.

For the data word address, the lower three bits are internally incremented following the receipt of each data word, while the higher bits retain the memory page row location. When the word address internally generated reaches the page boundary,

the following byte is placed at the beginning of the same page. If more than eight data words are transmitted to the EEPROM, the data word address will “roll over” and previous data will be overwritten.

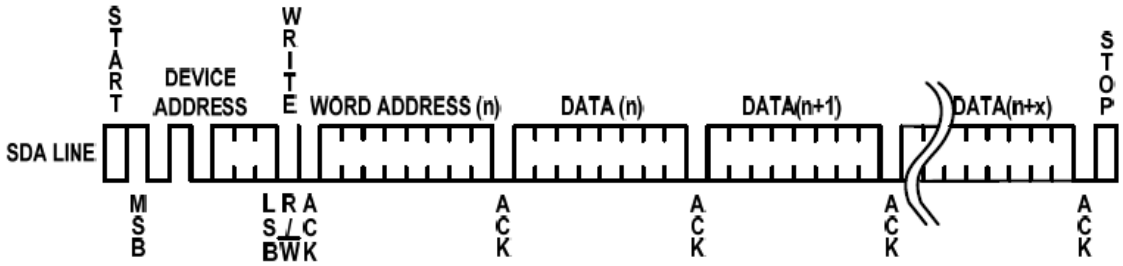



Figure 12- 6 Page write

Once the internally-timed write cycle is started and the EEPROM inputs are disabled, acknowledge polling can be initiated. This involves sending a start condition followed by the device address word. The R/W bit indicates read/write operation. Only if the internal write cycle is completed will the EEPROM respond with an ack “0” allowing the read/write sequence to continue.

### 12.2.2 Read operations

Read operation initiation is similar to write operation except that the R/W bit in the device address word should be set as “1”. Three read operations are supported, including current address read, random address read and sequential read.

For current address read, the internal data word address counter maintains the last address accessed during the last read/write operation, incremented by 1. This address stays valid between operations as long as the chip power is maintained. The address “roll over” during read is from the last byte of the last memory page to the first byte of the first page. The address “roll over” during write is from the last byte of the current page to the first byte of the same page. Once the device address with the R/W bit set as “1” is clocked in and acknowledged by the EEPROM, the data word of current address is serially clocked out. The MCU will generate a stop condition following NO

Shared under NL

ack.

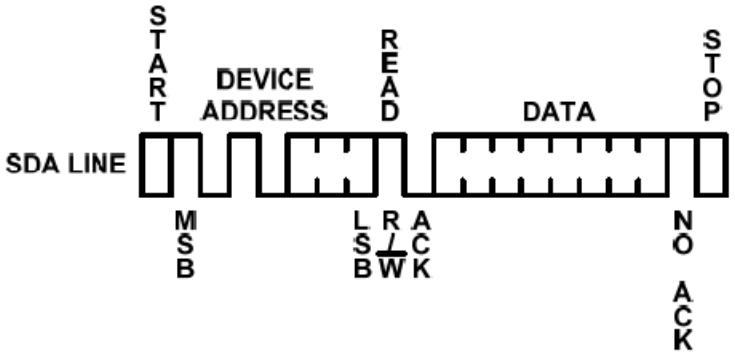



Figure 12- 7 Current address read

A random read requires a “dummy” byte write sequence to load in the data word address. Once the device address word and data word address are clocked in and acknowledged by the EEPROM, the MCU must generate another start condition. The MCU now initiates a current address read by sending a device address with the R/W bit set as “1”. The EEPROM acknowledges the device address and serially clocks out the data word. The MCU will generate a stop condition following NO ack.

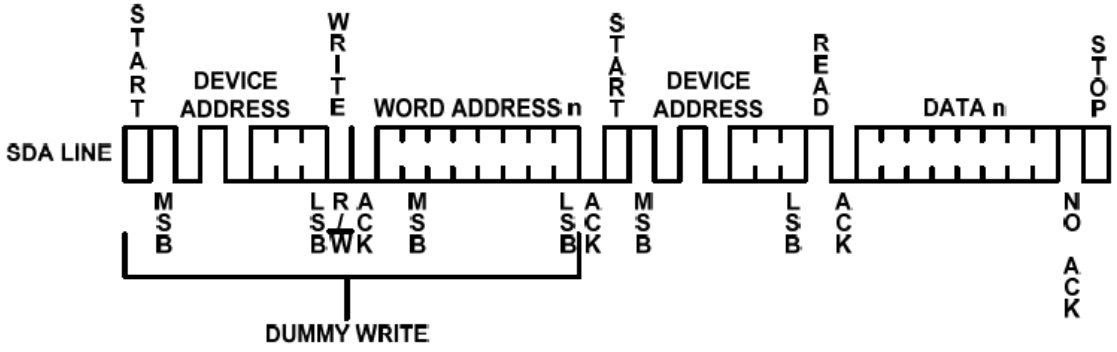



Figure 12- 8 Random read

Sequential read is initiated by either a current address read or a random address read. After the MCU receives a data word, it responds with an ack. As long as the EEPROM receives an ack, it will continue to increment the data word address and serially clock out sequential data words. When the memory address limit (2K) is reached, the data word address will “roll over” and the sequential read will continue. The

Strictly under NDA

sequential read operation is terminated when the MCU generates a stop condition following NO ack.

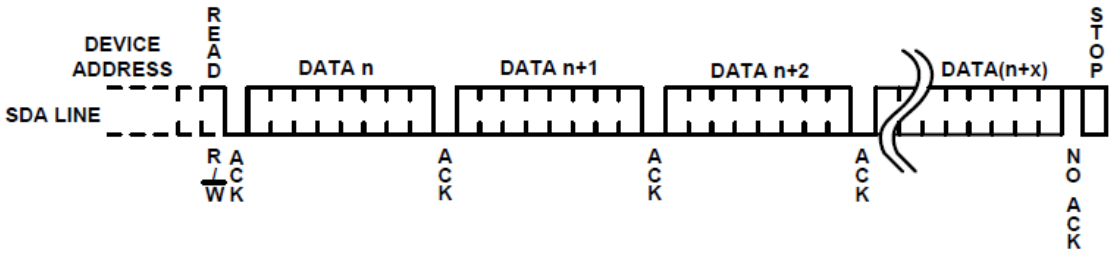



Figure 12- 9 Sequential read

Shared under NDA

# 13 AES

The TLSR8367 embeds AES module with encryption and decryption function. The input 128bit plaintext in combination of key is converted into the final output ciphertext via encryption; the 128bit ciphertext in combination of key can also be converted into 128bit plaintext via decryption.

The AES hardware accelerator provides automatic encryption and decryption. It only takes (1000*system clock cycles) to implement AES encryption/decryption. Suppose system clock is 20MHz, the time needed for AES encryption/decryption is 50us.

## 13.1 RISC mode

For RISC mode, configuration of related registers is as follows:

1) Set the value of key via writing registers AES_KEY0~ AES_KEY15 (address 0x550~0x55f).
2) Set operation method of AES module via register AES_CTRL: set address 0x540[0] as 1b'1 for decryption method, while clear this bit for encryption method.
3) For encryption method, write registers AES-DAT0~ AES-DAT3 (address 0x548~0x54b) for four times to set the 128bit plaintext. After encryption, the 128bit ciphertext can be obtained by reading address 0x548~0x54b for four times.
4) For decryption method, write registers AES-DAT0~ AES-DAT3 (address 0x548~0x54b) for four times to set the 128bit ciphertext. After decryption, the 128bit plaintext can be obtained by reading address 0x548~0x54b for four times.
5) Address 0x540 bit[1] and bit[2] are read only bits: bit[1] will be cleared automatically after quartic writing of address 0x548~0x54b; bit[2] will be set as 1 automatically after encryption/decryption, and then cleared automatically after quartic reading of address 0x548~0x54b.

## 13.2 AES-CCM

The AES-CCM (Counter with the CBC-MAC) mode is disabled by default. AES output is directly determined by current encryption and decryption, irrespective of previous encryption and decryption result.

If 0x540[7] is set as 1b'1 to enable AES-CCM mode, AES output will also take previous encryption and decryption result into consideration.

Shared under NDA

### 13.3 Register table

Table 13- 1 Register table related to AES

| Address | Mnemonic | Type | Description | Reset Value |
|---|---|---|---|---|
| 0x540 | AES_CTRL | R/W | [0] Select decrypt/encrypt.<br>1: decrypt, 0: encrypt<br>[1] Read-only.<br>1: input data needed,<br>0: input data ready.<br>[2] Read-only.<br>0: output data not ready,<br>1: output data ready.<br>[7] 1: enable AES-CCM mode. | 00 |
| 0x548 | AES-DAT0 | R/W | Input/Output Data byte 0 | *No power on reset |
| 0x549 | AES-DAT1 | R/W | Input/Output Data byte 1 | |
| 0x54a | AES-DAT2 | R/W | Input/Output Data byte 2 | |
| 0x54b | AES-DAT3 | R/W | Input/Output Data byte 3 | |
| 0x550 | AES_KEY0 | R/W | [7:0] KEY0 | 00 |
| 0x551 | AES_KEY1 | R/W | [7:0] KEY1 | 00 |
| 0x552 | AES_KEY2 | R/W | [7:0] KEY2 | 00 |
| 0x553 | AES_KEY3 | R/W | [7:0] KEY3 | 00 |
| 0x554 | AES_KEY4 | R/W | [7:0] KEY4 | 00 |
| 0x555 | AES_KEY5 | R/W | [7:0] KEY5 | 00 |
| 0x556 | AES_KEY6 | R/W | [7:0] KEY6 | 00 |
| 0x557 | AES_KEY7 | R/W | [7:0] KEY7 | 00 |
| 0x558 | AES_KEY8 | R/W | [7:0] KEY8 | 00 |
| 0x559 | AES_KEY9 | R/W | [7:0] KEY9 | 00 |
| 0x55a | AES_KEY10 | R/W | [7:0] KEY10 | 00 |
| 0x55b | AES_KEY11 | R/W | [7:0] KEY11 | 00 |
| 0x55c | AES_KEY12 | R/W | [7:0] KEY12 | 00 |
| 0x55d | AES_KEY13 | R/W | [7:0] KEY13 | 00 |
| 0x55e | AES_KEY14 | R/W | [7:0] KEY14 | 00 |
| 0x55f | AES_KEY15 | R/W | [7:0] KEY15 | 00 |

***Note:** Addresses 0x548~0x54b won't be reset after power on.

Shared under NDA

# 14 Key Electrical Specifications

**Note:** The electrical characteristics currently listed in this section are target specifications and only supplied for reference. Some data may be updated according to actual test results.

## 14.1 Absolute maximum ratings

Table 14- 1 Absolute Maximum Ratings

| Characteristics | Sym. | Min. | Max | Unit | Test Condition |
|---|---|---|---|---|---|
| Supply Voltage | VDD | -0.3 | 4.3 | V | All AVDD and DVDD pin must have the same voltage |
| Voltage on Input Pin | $V_{In}$ | -0.3 | VDD+ 0.3 | V | |
| Output Voltage | $V_{Out}$ | 0 | VDD | V | |
| Storage temperature Range | $T_{Str}$ | -65 | 150 | °C | |
| Soldering Temperature | $T_{Sld}$ | | 260 | °C | |

**CAUTION:** Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

## 14.2 Recommended operating condition

Table 14- 2 Recommended Operation Condition

| Item | Sym. | Min | Typ. | Max | Unit | Condition |
|---|---|---|---|---|---|---|
| Power-supply voltage | VDD | 1.9 | 3.3 | 4.3 | V | QFN package |
| | | 1.9 | 3.3 | 3.6 | V | SOP16 package[1] |
| Supply rise time (from 0V to 1.9V) | $t_R$ | | | 17.5[*2] | ms | |
| Operating Temperature Range | $T_{Opr}$ | -40 | | 85 | °C | |

[1] Note: By changing bonding wire, the SOP16 package can also support the power supply range of 1.9~4.3V.

[2] Note: For the rise time, if the reset pin cap is 1uF, the supply rise time to 1.9V is recommend to be smaller than 17.5ms as listed above. If larger cap is used, the supply rise time can increase proportionally. If the supply rise time exceeds the max allowed time, especially if the rise from the POR threshold in Section 14.4 to 1.9V is slow, the internal LDO may not settle to the desired states during this period and may lead to unexpected digital logic behavior or worsened RF performance. The rising slope below the POR threshold does not affect much and can be slower.

Shared under NDA

## 14.3 Electrical characteristics

Table 14- 3 Electrical characteristics

(Test condition: VDD=3.3V and T=25℃ for Typ. value)

| Mode | Sym. | Min | Typ. | Max | Unit | Condition |
|---|---|---|---|---|---|---|
| Tx | $I_{Tx}$ | | 14.5 | | mA | Continuous TX transmission, 0dBm output power |
| | | | 25 | | mA | maximum output power |
| Rx | $I_{Rx}$ | | 13.6 | | mA | Continuous Rx reception |
| Suspend | $I_{Susp}$ | | 6.8 | 20 | uA | IO wakeup |
| | | | 8 | 20 | uA | 32k RC wakeup |
| Deep sleep | $I_{Dps}$ | | 1.9 | | uA | internal 32kHz RC OSC off |
| | | | 3 | | uA | internal 32kHz RC OSC on |

## 14.4 General characteristics

Table 14- 4 General characteristics

| Item | Sym. | Min. | Typ. | Max. | Unit | Condition |
|---|---|---|---|---|---|---|
| **Power-up/ Power-down sequence (see section 2.6.1)** | | | | | | |
| VDD voltage when $V_{UVLO}$ turns to high level | $V_{POR}$ | | 1.70 | | V | |
| VDD voltage when $V_{UVLO}$ turns to low level | $V_{Pdn}$ | | 1.62 | | V | |
| Delay counter value | $T_{PwRst}$ | Configurable via analog register afe_0x20 | | | | |
| **Timing for working mode transition** (see section 2.4.3 Power-saving mode) | | | | | | |
| Transition time needed from suspend mode to the active mode | $T_{SP2A}$ | | 500 | | us | |
| Transition time needed from deep sleep mode with SRAM retention to active mode | $T_{DS2A}$ | | 500 | | us | |
| Transition time needed from deep sleep mode without SRAM retention to active mode | | | 1 | | ms | |

## 14.5 Inputs/Outputs characteristics

Shared under NDA

Table 14- 5 Inputs/Outputs Characteristics

| Item | Sym. | Min | Typ. | Max | Unit | Condition |
|---|---|---|---|---|---|---|
| **Digital inputs/outputs** | | | | | | |
| Input high voltage | VIH | 0.7VDD | | VDD | V | |
| Input low voltage | VIL | VSS | | 0.3VDD | V | |
| Output high voltage | VOH | 0.9VDD | | VDD | V | |
| Output low voltage | VOL | VSS | | 0.1VDD | V | |

## 14.6 Pull-up/Pull-down resistor

Table 14- 6 Pull-up/Pull-down resistor

(over process and T=-40~+85℃)

| Pull-up/Pull-down resistor | Min | Max | Condition |
|---|---|---|---|
| x1 pull-up | 8 | 11 | VDD=3.3~4.3V |
| | 10 | 12 | VDD=3V |
| | 16 | 19 | VDD=1.9V |
| x10 pull-down | 128 | 136 | VDD=3.3~4.3V |
| | 174 | 188 | VDD=3V |
| | 465 | 496 | VDD=1.9V |
| x100 pull-up | 756 | 785 | VDD=3.3~4.3V |
| | 953 | 1003 | VDD=3V |
| | 2011 | 2246 | VDD=1.9V |

Shared under NDA

### 14.7 SPI characteristics

Table 14- 7 SPI characteristics

(over process, voltage 1.9~4.3V, and T=-40~+85℃)

| Item | Sym. | Min | Typ. | Max | Unit | Condition |
|---|---|---|---|---|---|---|
| CK frequency | $F_{CK}$ | | | 4 | MHz | Slave |
| CK duty cycle clock | | | 50 | | % | Master |
| DI setup time | | 30 | | | ns | Slave |
| | | 90 | | | ns | Master |
| DI hold time | | 10 | | | ns | Slave |
| | | 90 | | | ns | Master |
| CK low to DO valid time | | | | 30 | ns | Slave |
| | | | | 120 | ns | Master |
| CN setup time | | 60 | | | ns | Master/Slave |
| CN high to DI tri-state*[3] | | | | | ns | Master |

Shared under NDA

[3] Note: Master actively stops reading during transmission, and Slave releases its driver DO and turns to tri-state.

### 14.8 I2C characteristics

Table 14- 8 I2C characteristics

(over process, voltage 1.9~4.3V, and T=-40~+85℃)

| Item | Sym. | Standard mode | | Fast mode | | Unit | Condition |
|---|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | | |
| SCL frequency | $F_{SCL}$ | | 100 | | 400 | kHz | |
| Rise time of SDA and SCL signals | $T_R$ | | 1000 | | 300 | ns | |
| Fall time of SDA and SCL signals | $T_F$ | | 300 | | 300 | ns | |
| START condition hold time | $T_{HD;STA}$ | 4 | | 0.6 | | us | |
| Data hold time | $T_{HD;DAT}$ | 0 | 3.45 | | 0.9 | us | |
| Data setup time | $T_{SU;DAT}$ | 250 | | 100 | | ns | |
| STOP condition setup time | $T_{SU;STO}$ | 4 | | 0.6 | | us | |

Shared under NDA

## 14.9 RF performance

Table 14- 9 RF characteristics

| Item | | Min | Typ. | Max | Unit | Condition |
|---|---|---|---|---|---|---|
| RF frequency range | | 2380 | | 2500 | MHz | Programmable in 1MHz step |
| Data rate | 2.4GHz 1Mbps, ±250kHz deviation<br>2.4GHz 2Mbps, ±500kHz deviation | | | | | |
| 2.4GHz 1Mbps RF_Rx performance | | | | | | |
| Sensitivity | 1Mbps | | -90 | | dBm | |
| Frequency Offset Tolerance | | -300 | | +300 | kHz | |
| Co-channel rejection | | | 5 | | dB | Wanted signal at -67dBm |
| In-band rejection C/I (modulation interference) | ±1MHz offset | | 0 | | dB | Wanted signal at -67dBm |
| | ±2 MHz offset | | -30 | | dB | |
| | ±3 MHz offset | | -34 | | dB | |
| | >4MHz offset | | -40 | | dB | |
| Image rejection | | | -34 | | dB | Wanted signal at -67dBm |
| 2.4GHz 1Mbps RF_Tx performance | | | | | | |
| Output power, maximum setting | | | 7 | | dBm | |
| Output power, minimum setting | | | -20 | | dBm | |
| Programmable output power range | | | 27 | | dB | |
| Modulation 20dB bandwidth | | | 1.5 | | MHz | |
| Adjacent channel power ratio (ACPR) | $F_0$±2MHz | | 57 | | dBc | |
| | $F_0$±4MHz | | 64 | | dBc | |
| | $F_0$±(>4MHz) | | 66 | | dBc | |
| Inband Emission | 2MHz | | -41 | | dBm | |
| | >=3MHz | | -42 | | dBm | |
| Spurious Emission Conducted Measurement | f < 1GHz | | -70 | | dBm | T=25 ℃, Max Output Power, Max Voltage |
| | f > 1GHz | | -68 | | dBm | |
| 2.4GHz 2Mbps RF_Rx performance | | | | | | |
| Sensitivity | 2Mbps | | -86.5 | | dBm | |
| Frequency Offset Tolerance | | -200 | | +200 | kHz | |
| Co-channel | | | 7 | | dB | Wanted signal |

shared under NDA

| Item | | Min | Typ. | Max | Unit | Condition |
|---|---|---|---|---|---|---|
| rejection | | | | | | at -67dBm |
| In-band rejection C/I (modulation interference) | ±2MHz offset | | -4 | | dB | Wanted signal at -67dBm |
| | ±4MHz offset | | -31 | | dB | |
| | ±6MHz offset | | -38 | | dB | |
| | ≥8MHz offset | | -42 | | dB | |
| Image rejection | | | -40 | | dB | Wanted signal at -67dBm |
| 2.4GHz 2Mbps RF_Tx performance | | | | | | |
| Output power, maximum setting | | | 7 | | dBm | |
| Output power, minimum setting | | | -20 | | dBm | |
| Programmable output power range | | | 27 | | dB | |
| Modulation 20dB bandwidth | | | 2.6 | | MHz | |
| Adjacent channel power ratio (ACPR) | $F_0$±4MHz | | 53 | | dBc | |
| | $F_0$±6MHz | | 63 | | dBc | |
| | $F_0$±(>6MHz) | | 62 | | dBc | |
| Spurious Emission Conducted Measurement | f < 1GHz | | -70 | | dBm | T=25 ℃, Max Output Power, Max Voltage |
| | f > 1GHz | | -68 | | dBm | |
| RSSI | | | | | | |
| RSSI range | | -88 | | -30 | dBm | |
| RSSI Resolution | | | +/-4 | | dB | |

Shared under NDA

## 14.10 Crystal characteristics

Table 14- 10 Crystal characteristics

(Test condition: VDD=3.3V, T=25℃)

| Item | Sym. | Min | Typ. | Max | Unit | Condition |
|---|---|---|---|---|---|---|
| **24MHz crystal** | | | | | | |
| Nominal frequency (parallel resonant) | $f_{NOM}$ | | 24 | | MHz | |
| Frequency tolerance | $f_{TOL}$ | -20 | | +20 | ppm | |
| Load capacitance | $C_L$ | 5 | 12 | 18 | pF | Programmable on chip load cap (PV @ 25℃ for min and Max) |
| Equivalent series resistance | ESR | | 40 | 80 | ohm | PVT |

## 14.11 RC oscillator characteristics

Table 14- 11 RC oscillator characteristics

(Test condition: VDD=3.3V, T=25℃)

| Item | Sym. | Min | Typ. | Max | Unit | Condition |
|---|---|---|---|---|---|---|
| **24MHz RC oscillator** | | | | | | |
| Nominal frequency | $f_{NOM}$ | | 24 | | MHz | |
| Frequency tolerance | $f_{TOL}$ | | 0.07 | | % | With calibration |
| **32kHz RC oscillator** | | | | | | |
| Nominal frequency | $f_{NOM}$ | | 32 | | kHz | |
| Frequency tolerance | $f_{TOL}$ | | 0.45 | | % | With calibration |
| Calibration time | | | 3 | | ms | |

## 14.12 ADC characteristics

Table 14- 12 ADC characteristics

(over process, voltage 1.9~3.6V, and T=-40~+85℃)

| Item | Sym. | Min | Typ. | Max | Unit | Condition |
|---|---|---|---|---|---|---|
| Differential nonlinearity | DNL | | | | LSB | |
| Integral nonlinearity | INL | | | | LSB | |
| Signal-to-noise ratio | SNR | | | | dB | fin=1kHz, fS=16kHz |
| Total harmonic distortion | THD | | | | dB | fin=628.66Hz, fS=100kHz |
| Effective Number of Bits | ENOB | | | | bits | |

Shared under NDA

| | | | | | | |
|---|---|---|---|---|---|---|
| Sampling frequency | Fs | | | | kHz | |

# 15 Application

*Note: Not all pin functions are listed in the reference design of this section.

## 15.1 Application example for TLSR8367EP16

### 15.1.1 Schematic

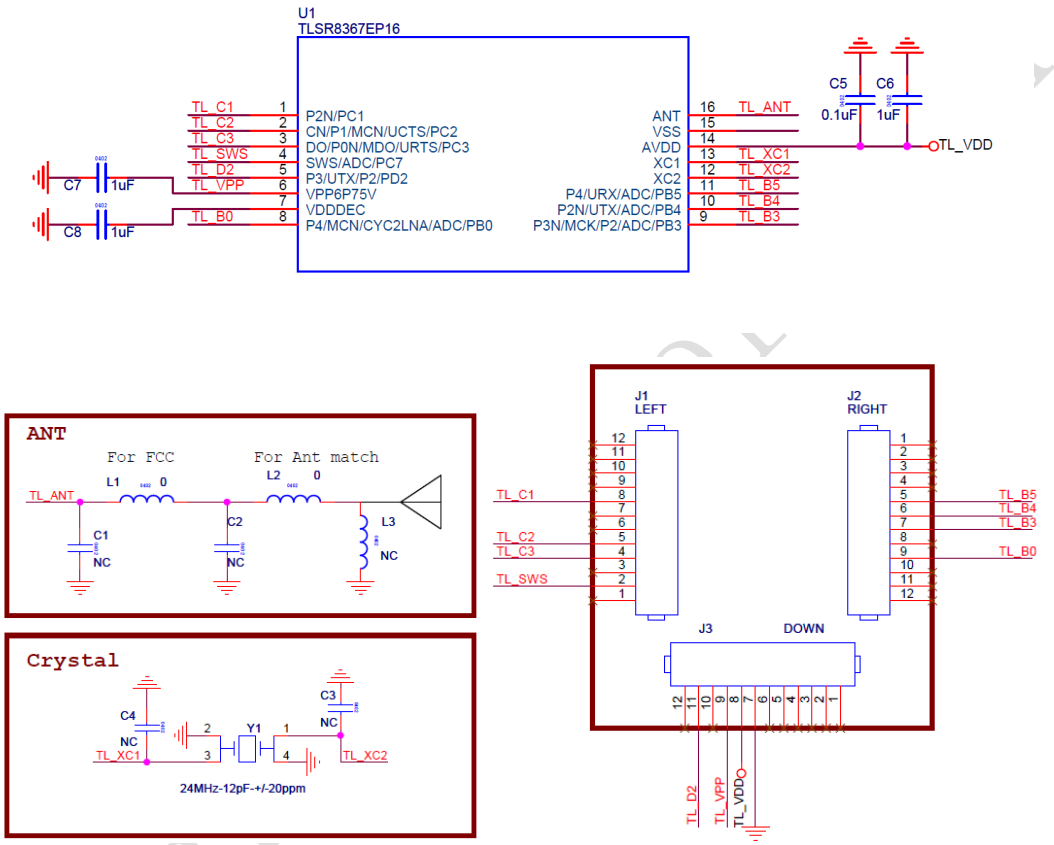



Figure 15- 1Schematic for TLSR8367EP16

### 15.1.2 BOM (Bill of Material)

Table 15- 1 BOM table for TLSR8367EP16

| Quantity | Reference | Description |
|---|---|---|
| **1** | C5 | 0.1uF |
| **3** | C6 | 1uF |
| | C7 | 1uF |
| | C8 | 1uF |
| **1** | J1 | LEFT |
| **1** | J2 | RIGHT |
| **1** | J3 | DOWN |
| **2** | L1 | 0 |
| | L2 | 0 |
| **1** | U1 | TLSR8367EP16 |
| **1** | Y1 | 24MHz-12pF-+/-20ppm |

Shared under NDA

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Telink:

TLSR8367F64ES16 TLSR8367EP16